Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«__» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека та захист інформації
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: «Інструмент тестування безпеки веб-додатків з використанням поліглотних векторів»

Виконавець: студент IV курсу, групи КБ-43

Дмитро МІЩЕНКО
(підпис) (ім’я прізвище)

	Ім’я, прізвище	**Підпис**
Керівник роботи	Олександр ТОРОШАНКО	

Нормоконтроль	Яніна ШЕСТАК	

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека та захист інформації
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої-професійної програми)

Студенту **КБ-43** (група) **Міщенку Дмитру Вадимовичу** (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Інструмент тестування безпеки веб-додатків з використанням поліглотних векторів

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Методика виявлення веб-вразливостей, архітектура системи виявлення вразливостей

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Аналіз сучасних типів вразливостей веб-додатків та методи їх виявлення.
Дослідження технології поліглотних векторів та обґрунтувати їх застосування для виявлення вразливостей веб-додатків.
Розробка програмного сканера веб-вразливостей.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Полягає у створенні ефективного інструменту для Аудиту безпеки веб-додатків.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняв до виконання _____ Дмитро МІЩЕНКО
(підпис) (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 20.01.2024	*виконано*
2	Аналіз літератури	21.01.2024 –20.02.2024	*виконано*
3	Дослідження поліглотних векторів	24.02.2024 – 04.03.2024	*виконано*
4	Дослідження сучасних типів вразливостей веб-додатків	05.03.2024 – 21.03.2024	*виконано*
5	Аналіз існуючих методів виявлення вразливостей веб-додатків	22.03.2024 – 09.04.2024	*виконано*
6	Вдосконалення індикаторів виявлення вразливостей	10.04.2024 – 15.04. 2024	*виконано*
7	Проектування архітектури системи виявлення вразливостей у веб-додатках	16.04.2024 – 22.04. 2024	*виконано*
8	Реалізація програмного сканера веб-вразливостей	23.04. 2024 – 30.04. 2024	*виконано*
9	Оцінка якості виявлення вразливостей розробленого сканера	1.05.2024 – 27.05.2024	*виконано*
10	Оформлення пояснювальної записки	28.05.2024 – 05.06.2024	*виконано*

Завдання видав _____ Олександр ТОРОШАНКО

	(підпис)	(ім’я, прізвище)
Завдання прийняла до виконання		Дмитро МІЩЕНКО
	(підпис)	(ім’я, прізвище)

Дата видачі завдання: 29.10.2024 р.

Термін подання кваліфікаційної роботи до ЕК 19.05.2025 р.

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку джерел, які були використані під час написання роботи, має 55 сторінок основного тексту, 5 рисунків, 4 таблиці. Список джерел, які були використані, містить 22 найменування та займає 4 сторінки.

Метою роботи є розробка та вдосконалення ефективної системи виявлення критичних вразливостей веб-додатків з використанням поліглотних векторів

Об’єктом дослідження є процеси виявлення та аналізу вразливостей веб-додатків.

Предметом дослідження є методи та технології виявлення критичних вразливостей веб-додатків з використанням поліглотних векторів та аналізу часу відповіді.

Методи дослідження:

- аналіз відкритих джерел
- аналіз сучасних типів вразливостей веб-додатків та існуючих методів їх виявлення
- реалізація програмного сканера веб-вразливостей

Практичне значення роботи полягає у створенні ефективного інструменту для аудиту безпеки веб-додатків, який може бути інтегрований у процеси DevSecOps та використаний фахівцями з інформаційної безпеки для підвищення захищеності корпоративних веб-систем.

Ключові слова: веб-безпека, виявлення вразливостей, поліглотні вектори, SQL-ін'єкції, XSS, аналіз часу відповіді, headless-браузер, кібербезпека.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕННЬ

OWASP – Open Web Application Security Project

XSS – Cross-Site Scripting

CSRF – Cross-Site Request Forgery

SAST – Static Application Security Testing

DAST – Dynamic Application Security Testing

RNN – Recurrent Neural Network

CNN – Convolutional Neural Network

LFI – Local File Inclusion

ORM – Object-Relational Mapping

ЗМІСТ

ВСТУП

Актуальність теми: розвиток інформаційних технологій і повсюдне впровадження веб-додатків у різноманітні сфери людської діяльності супроводжується зростанням кількості кібератак та виявленням нових типів вразливостей. За даними OWASP (Open Web Application Security Project), кількість критичних вразливостей у веб-додатках зростає щороку, а економічні збитки від успішних атак сягають мільярдів доларів. Особливу загрозу становлять такі вразливості як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та інші, що можуть призвести до витоку конфіденційних даних, фінансових втрат та репутаційних ризиків. Існуючі методи виявлення вразливостей часто не забезпечують необхідного рівня захисту через низьку ефективність, велику кількість хибно-позитивних результатів або обмежені можливості виявлення складних поліморфних атак. Тому розробка нових і вдосконалення існуючих методів виявлення вразливостей веб-додатків, зокрема з використанням поліглотних векторів та аналізу часу відповіді, є надзвичайно актуальним завданням для забезпечення інформаційної безпеки.

Мета і завдання дослідження: розробка та вдосконалення системи виявлення критичних вразливостей веб-додатків на основі поліглотних векторів та аналізу часу відповіді для підвищення ефективності захисту інформаційних ресурсів.

Завдання дослідження:

1. Проаналізувати сучасні типи вразливостей веб-додатків та існуючі методи їх виявлення для визначення основних напрямків вдосконалення.
2. Дослідити технологію поліглотних векторів та обґрунтувати їх застосування для виявлення вразливостей веб-додатків.
3. Розробити та вдосконалити методи аналізу часу відповіді як індикатора наявності вразливостей у веб-додатках.

4. Спроектувати архітектуру системи виявлення вразливостей та розробити алгоритми виявлення SQL-ін'єкцій, міжсайтового скриптингу та інших критичних вразливостей.

5. Реалізувати програмний сканер веб-вразливостей та оцінити його ефективність на реальних веб-додатках.

Об'єкт дослідження: процеси виявлення та аналізу вразливостей веб-додатків.

Предмет дослідження: Методи та технології виявлення критичних вразливостей веб-додатків з використанням поліглотних векторів та аналізу часу відповіді.

Практичне значення роботи: практичне значення роботи полягає у розробці ефективної системи виявлення критичних вразливостей веб-додатків, що може бути використана як інструмент для проведення аудиту безпеки та захисту інформаційних ресурсів організацій різного масштабу. Запропоновані методи виявлення SQL-ін'єкцій, міжсайтового скриптингу та інших вразливостей з використанням поліглотних векторів дозволяють значно знизити кількість хибно-позитивних результатів, а також виявляти складні поліморфні атаки, які можуть обходити традиційні системи захисту. Розроблений програмний сканер може бути інтегрований у процеси розробки веб-додатків за методологією DevSecOps, що забезпечить раннє виявлення вразливостей ще на етапі розробки. Отримані результати дослідження та практичні рекомендації можуть бути використані фахівцями з інформаційної безпеки для підвищення захищеності корпоративних веб-систем та зниження ризиків кібератак, а також включені до навчальних матеріалів з безпеки веб-додатків для підготовки спеціалістів у галузі кібербезпеки.

Методи дослідження*:* Розробка та вдосконалення системи виявлення критичних вразливостей веб-додатків на основі поліглотних векторів та аналізу часу відповіді для підвищення ефективності захисту інформаційних ресурсів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ

1.1 Аналіз сучасних типів вразливостей веб-додатків та методів їх виявлення

Вразливості веб-додатків представляють собою недоліки в програмному коді, конфігурації або бізнес-логіці, які можуть бути використані зловмисниками для отримання несанкціонованого доступу до інформації, порушення роботи системи або виконання інших неавторизованих дій. Згідно з дослідженнями Open Web Application Security Project (OWASP), найпоширенішими типами вразливостей веб-додатків є: ін'єкційні атаки (зокрема SQL-ін'єкції), некоректна автентифікація та управління сесіями, міжсайтовий скриптинг (XSS), небезпечні прямі посилання на об'єкти, некоректна конфігурація безпеки, витік конфіденційної інформації, відсутність функціонального рівня контролю доступу, міжсайтова підробка запитів (CSRF), використання компонентів з відомими вразливостями та незахищені перенаправлення. Кожен із цих типів вразливостей має свої особливості, вектори атаки та потенційні наслідки для безпеки інформаційних систем. Наприклад, SQL-ін'єкції дозволяють зловмисникам виконувати довільні SQL-запити до бази даних, що може призвести до витоку, зміни або видалення даних, тоді як XSS-атаки надають можливість виконувати шкідливий JavaScript-код у браузері користувача, що може бути використано для крадіжки сесійних токенів, перехоплення паролів або виконання дій від імені користувача [1].

Методи виявлення вразливостей веб-додатків можна умовно поділити на статичний та динамічний аналіз коду, а також тестування на проникнення (penetration testing). Статичний аналіз безпеки додатків (SAST) передбачає дослідження вихідного коду без його виконання, з метою виявлення потенційних вразливостей. Цей метод дозволяє знаходити вразливості ще на

етапі розробки, до того, як код потрапить у виробниче середовище. Для проведення статичного аналізу використовуються спеціалізовані інструменти, які перевіряють код на відповідність правилам безпечного програмування та шукають типові патерни вразливостей. Динамічний аналіз безпеки додатків (DAST) здійснюється шляхом тестування працюючого додатку та імітації атак на нього. Цей підхід дозволяє виявляти вразливості, які можуть бути непомітними при статичному аналізі, оскільки вони можуть виникати внаслідок взаємодії різних компонентів системи або специфічних умов середовища виконання. Тестування на проникнення є комплексним підходом, який поєднує автоматизовані інструменти та ручні методи для моделювання реальних атак на веб-додаток з метою виявлення вразливостей, які можуть бути експлуатовані зловмисниками.

Сучасні методи виявлення SQL-ін'єкцій базуються на різноманітних підходах, включаючи аналіз сигнатур, аналіз аномалій, тайнтінг (taint analysis) та машинне навчання. Аналіз сигнатур передбачає пошук відомих патернів атак у вхідних даних, однак цей метод не ефективний проти нових або модифікованих типів атак. Аналіз аномалій фокусується на виявленні відхилень від нормальної поведінки додатку, що дозволяє виявляти невідомі раніше типи атак, але може призводити до значної кількості хибно-позитивних результатів. Тайнтінг відстежує поширення небезпечних даних через додаток, від джерела (вхідні дані) до приймача (наприклад, SQL-запиту), що дозволяє виявляти вразливості, пов'язані з недостатньою валідацією або санітизацією вхідних даних. Методи на основі машинного навчання використовують різноманітні алгоритми для класифікації запитів як шкідливих або безпечних на основі виділених характеристик, таких як наявність специфічних SQL-ключових слів, довжина запиту, ентропія тощо. Хоча жоден з цих методів не забезпечує стовідсоткового захисту, їх комбінація дозволяє досягти високого рівня виявлення SQL-ін'єкцій у веб-додатках [2].

Міжсайтовий скриптинг (XSS) залишається одним з найпоширеніших типів вразливостей веб-додатків, що дозволяє зловмисникам впроваджувати шкідливий код у веб-сторінки, які переглядаються іншими користувачами.

Виділяють три основні типи XSS-атак: відображені (reflected), збережені (stored) та DOM-based. Відображені XSS-атаки виникають, коли шкідливий код, надісланий зловмисником у запиті, відображається на сторінці без належної санітизації. Збережені XSS-атаки передбачають збереження шкідливого коду в базі даних або іншому сховищі даних, з подальшим його відображенням для всіх користувачів, які переглядають відповідну сторінку. DOM-based XSS-атаки використовують вразливості в JavaScript-коді, який модифікує DOM-структуру сторінки на стороні клієнта. Методи виявлення XSS-вразливостей включають статичний та динамічний аналіз коду, фаззінг (fuzzing) — метод тестування, який передбачає подачу на вхід програми неочікуваних або некоректних даних, а також використання спеціалізованих сканерів безпеки, які автоматично перевіряють веб-додатки на наявність типових XSS-вразливостей. Ефективність цих методів залежить від багатьох факторів, включаючи складність додатку, використані технології та якість реалізації механізмів захисту.

Сучасні тенденції в області виявлення вразливостей веб-додатків включають використання інтелектуальних систем, які поєднують традиційні методи з технологіями штучного інтелекту та машинного навчання. Такі системи здатні аналізувати великі обсяги даних, виявляти складні паттерни атак та адаптуватися до нових загроз. Зокрема, рекурентні нейронні мережі (RNN) та згорткові нейронні мережі (CNN) успішно застосовуються для класифікації запитів як шкідливих або безпечних на основі послідовностей символів або інших характеристик. Глибоке навчання з підкріпленням (Deep Reinforcement Learning) дозволяє системам безпеки автоматично знаходити оптимальні стратегії для виявлення та запобігання атакам на веб-додатки. Ансамблеві методи, які комбінують результати різних алгоритмів машинного навчання, демонструють підвищену точність виявлення вразливостей порівняно з окремими методами. Незважаючи на значний прогрес у цій галузі, залишаються невирішеними питання інтерпретованості результатів, отриманих за допомогою моделей машинного навчання, а також проблеми, пов'язані з можливістю обходу таких систем зловмисниками через застосування спеціальних технік ухилення [3, с. 57].

Інтеграція методів виявлення вразливостей у процеси розробки веб-додатків відповідно до концепції DevSecOps є надзвичайно перспективним напрямком, який дозволяє забезпечити безпеку на всіх етапах життєвого циклу програмного забезпечення. Впровадження автоматизованих інструментів аналізу безпеки у конвеєри безперервної інтеграції та доставки (CI/CD) дозволяє виявляти вразливості ще до того, як код потрапить у виробниче середовище, що значно знижує ризики та вартість виправлення проблем. Використання фреймворків безпечної розробки (Secure Development Frameworks) та бібліотек, які забезпечують вбудовані механізми захисту від поширених типів атак, таких як об'єктно-реляційні мапери (ORM) для запобігання SQL-ін'єкціям або шаблонізатори з автоматичною санітизацією для запобігання XSS-атакам, також сприяє підвищенню рівня безпеки веб-додатків. Регулярні аудити безпеки, проведені кваліфікованими фахівцями, залишаються невід'ємною частиною комплексного підходу до виявлення вразливостей, оскільки дозволяють виявляти складні вразливості, які можуть бути пропущені автоматизованими інструментами. Таким чином, ефективна стратегія виявлення вразливостей веб-додатків повинна включати комбінацію різних методів та інструментів, адаптованих до конкретного контексту розробки та експлуатації додатку, а також враховувати постійний розвиток ландшафту загроз та технологій [4, с. 115].

Важливим аспектом програмної реалізації сканера веб-вразливостей є розробка системи обробки та аналізу результатів сканування, яка забезпечує ефективну інтерпретацію отриманих даних та формування звітів у зручному форматі. Цей компонент системи відповідає за збір та структурування інформації про виявлені вразливості, включаючи їх тип, URL-адресу, параметр, який є вразливим, використане навантаження та інші деталі, які необхідні для розуміння та усунення проблеми. Для зберігання результатів використовується структура даних vulns, яка представляє собою словник з ключами для кожного типу вразливості (rce, xss, sql, lfi) та значеннями, які представляють кількість виявлених вразливостей цього типу. Крім того, структура включає ключ list, значення якого є рядком з детальним описом кожної виявленої вразливості у

форматі ТИП|TYPE|URL:ПАРАМЕТР:НАВАНТАЖЕННЯ|DELIMITER|. Такий формат дозволяє легко парсити та візуалізувати результати на стороні клієнта, а також забезпечує можливість експорту даних у різні формати для подальшого аналізу. Результати сканування передаються клієнту у форматі JSON, який є стандартом для обміну даними у веб-середовищі та може бути легко інтегрований з іншими інструментами та системами.

Особливу увагу при реалізації сканера було приділено обробці помилок та виняткових ситуацій, які можуть виникати під час сканування веб-додатків. Система включає робастні механізми обробки винятків, які забезпечують стабільну роботу навіть при непередбачуваній поведінці цільового додатку або мережевих проблемах. Наприклад, у функції scan_xss реалізовано механізм, який дозволяє виявляти XSS-вразливості навіть у випадках, коли headless-браузер Ghost не може правильно обробити відповідь сервера. Якщо в тексті винятку присутнє слово "confirm" або "alert", це може свідчити про успішну XSS-атаку, навіть якщо браузер не зміг коректно виконати JavaScript-код. Аналогічно, в інших функціях сканування реалізовані механізми для обробки типових помилок, таких як таймаути з'єднання, помилки HTTP, некоректні відповіді сервера та інші. Це забезпечує високу надійність системи при скануванні різних типів веб-додатків, включаючи складні та нестандартні реалізації. Крім того, система включає механізми для обмеження навантаження на цільовий сервер, такі як затримки між запитами та обмеження кількості паралельних з'єднань, що дозволяє запобігти надмірному навантаженню на цільовий ресурс та потенційному відмову в обслуговуванні [2].

1.2 Дослідження технології поліглотних векторів та їх застосування для виявлення вразливостей

Поліглотні вектори представляють собою спеціалізований тип навантажень (payload), які одночасно є валідними в декількох контекстах або мовах програмування, що дозволяє їм обходити стандартні механізми захисту веб-додатків. Термін "поліглот" у контексті кібербезпеки означає конструкцію,

яка може бути інтерпретована різними способами в залежності від середовища виконання, що робить такі вектори особливо ефективними для виявлення складних вразливостей. Наприклад, поліглотний вектор може одночасно бути валідним HTML, JavaScript та SQL-кодом, що дозволяє йому проникати через різні рівні захисту та перевірки. Концепція поліглотних векторів базується на глибокому розумінні синтаксичних особливостей різних мов програмування та протоколів, а також на використанні цих особливостей для створення послідовностей символів, які будуть по-різному інтерпретовані на різних рівнях обробки даних. У контексті виявлення вразливостей веб-додатків, поліглотні вектори дозволяють ідентифікувати слабкі місця, які можуть бути пропущені традиційними методами аналізу безпеки, оскільки вони здатні експлуатувати ситуації, коли вхідні дані проходять через декілька рівнів обробки з різними правилами валідації та санітизації [5, с. 55].

Механізми функціонування поліглотних векторів засновані на використанні особливостей парсингу та інтерпретації різних мов програмування. Наприклад, символи, які є коментарями в одній мові, можуть мати синтаксичне значення в іншій, що дозволяє створювати конструкції, які будуть по-різному оброблятися різними компонентами системи. Крім того, багато мов програмування та протоколів мають механізми екранування спеціальних символів, які можуть бути використані для створення поліглотних конструкцій. Наприклад, послідовність символів /*<script>*/alert('XSS');//</script> може бути одночасно валідним коментарем в SQL та JavaScript-кодом, який буде виконаний у браузері користувача при відображенні результатів запиту. Така дуальність інтерпретації є основою для створення ефективних поліглотних векторів, які можуть проникати через різні рівні захисту. При розробці поліглотних векторів для виявлення вразливостей веб-додатків необхідно враховувати особливості взаємодії різних компонентів системи, включаючи клієнтський браузер, серверні скрипти, бази даних та інші елементи інфраструктури, які можуть бути задіяні в обробці та відображенні даних.

Процес створення та використання поліглотних векторів для виявлення вразливостей включає декілька етапів, від аналізу цільової системи до розробки специфічних конструкцій, які можуть експлуатувати виявлені слабкі місця. На першому етапі проводиться глибокий аналіз архітектури додатку, використаних технологій та механізмів захисту, що дозволяє визначити потенційні вектори атаки. Наступним кроком є розробка поліглотних конструкцій, які можуть обходити виявлені механізми захисту, використовуючи особливості різних мов програмування та протоколів. Третій етап передбачає тестування розроблених векторів на цільовій системі для виявлення вразливостей та оцінки їх експлуатованості. Нарешті, результати тестування аналізуються для визначення фактичного рівня ризику та розробки рекомендацій щодо усунення виявлених вразливостей. Цей процес вимагає глибоких знань в області веб-технологій, мов програмування та механізмів безпеки, а також здатності мислити нестандартно, щоб створювати конструкції, які можуть обійти стандартні механізми захисту [6].

Застосування поліглотних векторів для виявлення SQL-ін'єкцій демонструє значні переваги порівняно з традиційними методами. Класичні підходи до тестування на SQL-ін'єкції часто використовують прості патерни, такі як додавання лапок або SQL-ключових слів до вхідних даних, що може бути легко виявлено та заблоковано системами захисту. Поліглотні вектори, з іншого боку, дозволяють створювати конструкції, які можуть обходити механізми валідації, залишаючись валідними в різних контекстах. Наприклад, послідовність символів 1'; SELECT * FROM users; -- - може бути валідною як частина числового виразу, що дозволяє їй пройти валідацію на стороні клієнта, і одночасно викликати виконання додаткового SQL-запиту на сервері, якщо вхідні дані не валідуються належним чином. Більш складні поліглотні вектори можуть включати кодування, коментарі та інші техніки, які дозволяють обходити багаторівневі системи захисту. У таблиці 1.1 представлено порівняння ефективності традиційних та поліглотних векторів для виявлення SQL-ін'єкцій на основі експериментальних даних, отриманих при тестуванні різних типів веб-додатків [7].

Таблиця 1.1

Порівняння ефективності традиційних та поліглотних векторів для виявлення SQL-ін'єкцій

Тип вектора	Відсоток виявлених вразливостей	Швидкість виявлення	Рівень хибно-позитивних результатів
Традиційні вектори	68%	Висока	12%
Поліглотні вектори	89%	Середня	5%
Гібридний підхід	94%	Низька	3%

Поліглотні вектори також мають суттєві переваги при виявленні вразливостей міжсайтового скриптингу (XSS), особливо при роботі з сучасними веб-додатками, які використовують складні механізми санітизації вхідних даних. Традиційні XSS-вектори, такі як <script>alert('XSS')</script>, легко виявляються та блокуються більшістю систем захисту. Однак, поліглотні вектори можуть використовувати складні конструкції, які обходять стандартні механізми санітизації, залишаючись валідними в різних контекстах. Наприклад, послідовність символів "onerror="alert(1)" може бути частиною валідного атрибуту HTML-елемента та одночасно викликати виконання JavaScript-коду при виникненні помилки. Більш складні поліглотні вектори можуть включати кодування символів, використання нестандартних атрибутів, CSS-ін'єкції та інші техніки, які дозволяють обходити механізми захисту від XSS. Використання поліглотних векторів для виявлення XSS-вразливостей вимагає глибокого розуміння моделі DOM, механізмів обробки HTML та JavaScript у різних браузерах, а також знання різноманітних технік обходу стандартних механізмів захисту.

Перспективи розвитку технології поліглотних векторів для виявлення вразливостей веб-додатків пов'язані з автоматизацією процесу їх створення та використання, а також з розширенням їх застосування для виявлення нових типів вразливостей. Сучасні дослідження в цій області фокусуються на розробці алгоритмів машинного навчання, які здатні автоматично генерувати поліглотні вектори на основі аналізу цільової системи та відомих патернів вразливостей. Такі алгоритми можуть використовувати методи генетичного програмування, нейронні мережі та інші техніки для створення ефективних поліглотних конструкцій, які здатні обходити складні механізми захисту. Крім того, перспективним напрямком є інтеграція технології поліглотних векторів у автоматизовані системи тестування безпеки, що дозволить проводити більш глибокий та ефективний аналіз веб-додатків на наявність вразливостей. Подальший розвиток цієї технології також передбачає розширення її застосування для виявлення вразливостей у мобільних додатках, інтернеті речей (IoT) та інших сучасних технологічних платформах, які можуть бути вразливими до схожих типів атак. Таким чином, технологія поліглотних векторів залишається потужним інструментом для виявлення вразливостей у сучасних веб-додатках та має значний потенціал для подальшого розвитку та вдосконалення [8].

1.3 Методи аналізу часу відповіді як індикатора наявності вразливостей у веб-додатках

Аналіз часу відповіді веб-додатку представляє собою метод виявлення вразливостей, заснований на вимірюванні та інтерпретації часових інтервалів між надсиланням запиту та отриманням відповіді від сервера. Даний підхід базується на принципі, що наявність певних типів вразливостей може призводити до статистично значущих відхилень у часі обробки запитів. У контексті безпеки веб-додатків, час відповіді розглядається як непрямий канал витоку інформації (side-channel), який може розкривати деталі внутрішньої обробки даних та логіки додатку. Основна ідея полягає в тому, що час,

необхідний для обробки запиту, може варіюватися залежно від того, як саме оброблюються вхідні дані, чи виконуються додаткові операції, чи відбуваються помилки обробки, які згодом обробляються системою. Наприклад, запит, який викликає виконання додаткового SQL-запиту через успішну ін'єкцію, займатиме більше часу, ніж запит, який не викликає таких додаткових операцій. Ця часова різниця, хоч і невелика в окремих випадках, може бути статистично значущою при проведенні багаторазових вимірювань та застосуванні відповідних методів аналізу даних. Таким чином, аналіз часу відповіді дозволяє виявляти вразливості навіть у ситуаціях, коли безпосередній аналіз відповіді сервера не дає достатньо інформації для визначення наявності проблеми безпеки [9].

Методологія аналізу часу відповіді для виявлення SQL-ін'єкцій базується на вимірюванні часових затримок при виконанні спеціально сформованих запитів, які можуть викликати різну поведінку системи в залежності від наявності вразливості. Одним із основних підходів є використання булевих запитів з умовними конструкціями, які можуть викликати затримку виконання SQL-запиту при виконанні певної умови. Наприклад, конструкція ' OR IF(condition, SLEEP(5), 0) -- - може викликати затримку в 5 секунд, якщо умова condition виконується, що дозволяє дізнатися результат виконання цієї умови через аналіз часу відповіді. Це є основою для таких технік, як blind SQL injection та time-based SQL injection, які дозволяють отримувати інформацію з бази даних по одному біту, навіть якщо результати запиту не відображаються безпосередньо в відповіді сервера. Даний підхід особливо ефективний для виявлення сліпих (blind) SQL-ін'єкцій, при яких результати запиту не повертаються безпосередньо користувачу, а лише впливають на внутрішню логіку додатку. Для підвищення точності таких вимірювань застосовуються статистичні методи, включаючи багаторазові повторення запитів, фільтрацію аномальних значень та кореляційний аналіз отриманих результатів, що дозволяє мінімізувати вплив випадкових факторів, таких як мережеві затримки або навантаження на сервер [10].

Техніки аналізу часу відповіді для виявлення вразливостей поділяються на декілька категорій залежно від механізму формування затримки та методів

інтерпретації результатів. Перша категорія — це техніки, засновані на прямій маніпуляції часом виконання запиту через використання спеціальних функцій, таких як SLEEP() в MySQL, pg_sleep() в PostgreSQL або WAITFOR DELAY в MS SQL Server. Друга категорія включає техніки, засновані на непрямому впливі на час виконання запиту через виконання ресурсоємних операцій, таких як обробка великих обсягів даних, складні математичні обчислення або рекурсивні запити. Третя категорія — це техніки, засновані на аналізі природньої варіації часу відповіді без штучного впливу, які використовують статистичний аналіз для виявлення аномалій, що можуть свідчити про наявність вразливостей. Кожна з цих категорій має свої переваги та обмеження, і їх ефективність залежить від конкретних умов, таких як тип бази даних, конфігурація сервера, наявність механізмів захисту та інші фактори. В таблиці 1.2 наведено порівняння різних технік аналізу часу відповіді за їх основними характеристиками, що дозволяє вибрати найбільш відповідний підхід для конкретного сценарію тестування [11].

Таблиця 1.2

Порівняння технік аналізу часу відповіді для виявлення вразливостей

Техніка	Точність	Швидкість	Стійкість до шуму	Сфера застосування
Пряма маніпуляція часом	Висока	Висока	Висока	SQL-ін'єкції, XXE
Непряма маніпуляція часом	Середня	Низька	Середня	SQL-ін'єкції, XPath-ін'єкції
Статистичний аналіз варіацій	Низька	Середня	Низька	Будь-які типи ін'єкцій
Диференційний аналіз часу	Висока	Низька	Висока	Криптографічні вразливості
Паралельний аналіз часу	Середня	Висока	Середня	Розподілені системи

Методи аналізу часу відповіді мають особливе значення для виявлення вразливостей типу "сліпа ін'єкція" (blind injection), коли результати атаки не відображаються безпосередньо у відповіді сервера. В таких ситуаціях традиційні методи, засновані на аналізі вмісту відповіді, можуть бути неефективними, оскільки сервер не надає прямої інформації про успішність

ін'єкції. Концепція сліпої ін'єкції включає два основних підтипи: булеву сліпу ін'єкцію (boolean-based blind injection), при якій результат запиту впливає на логіку роботи додатку та може бути визначений через аналіз відмінностей у відповіді, та часову сліпу ін'єкцію (time-based blind injection), при якій результат запиту визначається через аналіз часу відповіді. Останній підтип є особливо складним для виявлення традиційними методами, оскільки відповідь сервера може бути ідентичною при наявності та відсутності вразливості, і лише час обробки запиту дає інформацію про успішність ін'єкції. Для ефективного виявлення таких вразливостей застосовуються специфічні техніки формування запитів, які максимізують часову різницю між позитивним та негативним сценаріями. Наприклад, можуть використовуватися конструкції, які викликають значну затримку при успішній ін'єкції, такі як рекурсивні запити, що обробляють великі обсяги даних, або виклики функцій затримки з великими значеннями параметрів. Додатково, для підвищення надійності результатів використовуються бінарний пошук, кодування даних та інші техніки, які дозволяють мінімізувати кількість необхідних запитів та підвищити точність аналізу.

Статистичні методи відіграють суттєву роль у підвищенні точності та надійності аналізу часу відповіді для виявлення вразливостей. Оскільки час відповіді може варіюватися під впливом різних факторів, таких як мережеві затримки, навантаження на сервер, кешування даних та інші, простий аналіз окремих вимірювань може призводити до хибних висновків. Для подолання цих обмежень застосовуються різноманітні статистичні підходи, включаючи методи фільтрації викидів, такі як Z-score або модифікований IQR (Interquartile Range), які дозволяють виявляти та виключати аномальні значення, що можуть спотворювати результати аналізу. Крім того, використовуються методи статистичної перевірки гіпотез, такі як t-тест або тест Манна-Уітні, для визначення статистичної значущості спостережуваних відмінностей у часі відповіді. Для врахування можливих змін у характеристиках системи з часом застосовуються методи аналізу часових рядів, такі як ковзне середнє або експоненціальне згладжування, які дозволяють адаптуватися до таких змін.

Інтеграція цих статистичних методів у системи виявлення вразливостей дозволяє значно підвищити точність та надійність результатів, зменшуючи кількість хибних спрацьовувань та пропущених вразливостей, що особливо актуально при роботі з складними веб-додатками в умовах нестабільного мережевого з'єднання або високого навантаження [12].

Перспективними напрямками розвитку методів аналізу часу відповіді є інтеграція з технологіями машинного навчання та штучного інтелекту, що дозволяє підвищити ефективність виявлення вразливостей у складних та динамічних середовищах. Сучасні дослідження в цій області фокусуються на застосуванні алгоритмів кластеризації, таких як K-means або DBSCAN, для автоматичного групування запитів за характеристиками часу відповіді, що дозволяє виявляти аномалії, які можуть свідчити про наявність вразливостей. Крім того, використовуються алгоритми класифікації, такі як Random Forest або Support Vector Machines, для побудови моделей, здатних визначати наявність вразливостей на основі часових характеристик відповіді та інших параметрів запиту. Особливу увагу дослідники приділяють методам навчання з підкріпленням (reinforcement learning), які дозволяють автоматично адаптувати стратегію тестування в процесі взаємодії з веб-додатком, оптимізуючи процес виявлення вразливостей. Також перспективним напрямком є розробка гібридних підходів, які комбінують аналіз часу відповіді з іншими методами, такими як аналіз вмісту відповіді, аналіз поведінки системи та статичний аналіз коду, що дозволяє отримати більш повну картину безпеки веб-додатку. Інтеграція цих передових технологій у системи виявлення вразливостей відкриває нові можливості для підвищення рівня захищеності веб-додатків, що особливо суттєво в умовах постійного ускладнення веб-технологій та зростання кількості та складності атак [11].

Сучасні тенденції в розвитку методів аналізу часу відповіді включають використання контейнерної віртуалізації та хмарних технологій для підвищення ефективності та масштабованості процесу виявлення вразливостей. Контейнерна віртуалізація, така як Docker та Kubernetes, дозволяє створювати ізольовані середовища для тестування веб-додатків, що забезпечує

стандартизацію та відтворюваність результатів незалежно від конкретного середовища розгортання. Це особливо суттєво для методів аналізу часу відповіді, які чутливі до різних факторів середовища, таких як навантаження на сервер, мережеві затримки та конфігурація системи. Хмарні технології, у свою чергу, надають можливість динамічно масштабувати ресурси для проведення великомасштабних тестів на наявність вразливостей, а також забезпечують доступ до розподілених обчислювальних ресурсів, що дозволяє проводити тестування з різних географічних локацій для врахування впливу мережевих затримок на результати аналізу. Крім того, використання хмарних сервісів для аналізу даних, таких як Amazon SageMaker, Google AI Platform або Microsoft Azure Machine Learning, дозволяє ефективно обробляти великі обсяги даних, отриманих у процесі тестування, та застосовувати складні алгоритми машинного навчання для підвищення точності виявлення вразливостей [13].

Етичні та правові аспекти використання методів аналізу часу відповіді для виявлення вразливостей веб-додатків є невід'ємною частиною дослідження цієї технології. Оскільки ці методи можуть бути використані як для захисту, так і для несанкціонованого доступу до інформаційних систем, існує потреба у встановленні чітких етичних принципів та правових норм, які регулюють їх застосування. Зокрема, необхідно забезпечити отримання належних дозволів перед проведенням тестування на наявність вразливостей, дотримуватися принципу мінімального втручання, який передбачає використання методів, що не призводять до порушення нормальної роботи системи або втрати даних, а також забезпечувати конфіденційність інформації, отриманої в процесі тестування. З правової точки зору, використання методів аналізу часу відповіді може підпадати під дію різних законодавчих актів, таких як Закон України "Про основні засади забезпечення кібербезпеки України", Конвенція про кіберзлочинність, Загальний регламент про захист даних (GDPR) та інші, які встановлюють відповідальність за несанкціонований доступ до інформаційних систем та вимоги щодо захисту персональних даних. Тому розробка та впровадження методів аналізу часу відповіді повинні здійснюватися з урахуванням відповідних етичних та правових норм, що забезпечить їх законне

та відповідальне використання для підвищення рівня безпеки інформаційних систем.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ВИЯВЛЕННЯ КРИТИЧНИХ ВРАЗЛИВОСТЕЙ

2.1 Архітектура системи виявлення вразливостей веб-додатків

Архітектура системи виявлення вразливостей веб-додатків представляє собою комплексну структуру взаємопов'язаних компонентів, які забезпечують ефективне виявлення та аналіз потенційних загроз безпеці. Запропонована архітектура базується на модульному підході, що дозволяє гнучко інтегрувати різні методи та техніки виявлення вразливостей в єдину систему. Основним елементом архітектури є серверний компонент, реалізований за допомогою фреймворку Flask, який забезпечує обробку запитів та координацію роботи всіх модулів системи. Серверний компонент складається з маршрутизатора запитів, який визначає порядок обробки вхідних даних та викликає відповідні модулі сканування. Головний маршрут системи ('/' в server.py) приймає параметри для сканування, включаючи URL-адресу цільового веб-додатку, користувацький агент, метод запиту, дані для передачі та інші необхідні параметри. Модуль обробки параметрів аналізує вхідні дані, визначає метод запиту (GET або POST), виділяє параметри для тестування та готує контекст для проведення сканування. Така структура дозволяє системі адаптуватися до різних типів веб-додатків та забезпечує ефективне використання ресурсів при проведенні сканування, оскільки кожен параметр аналізується окремо з використанням спеціалізованих модулів для різних типів вразливостей [14].

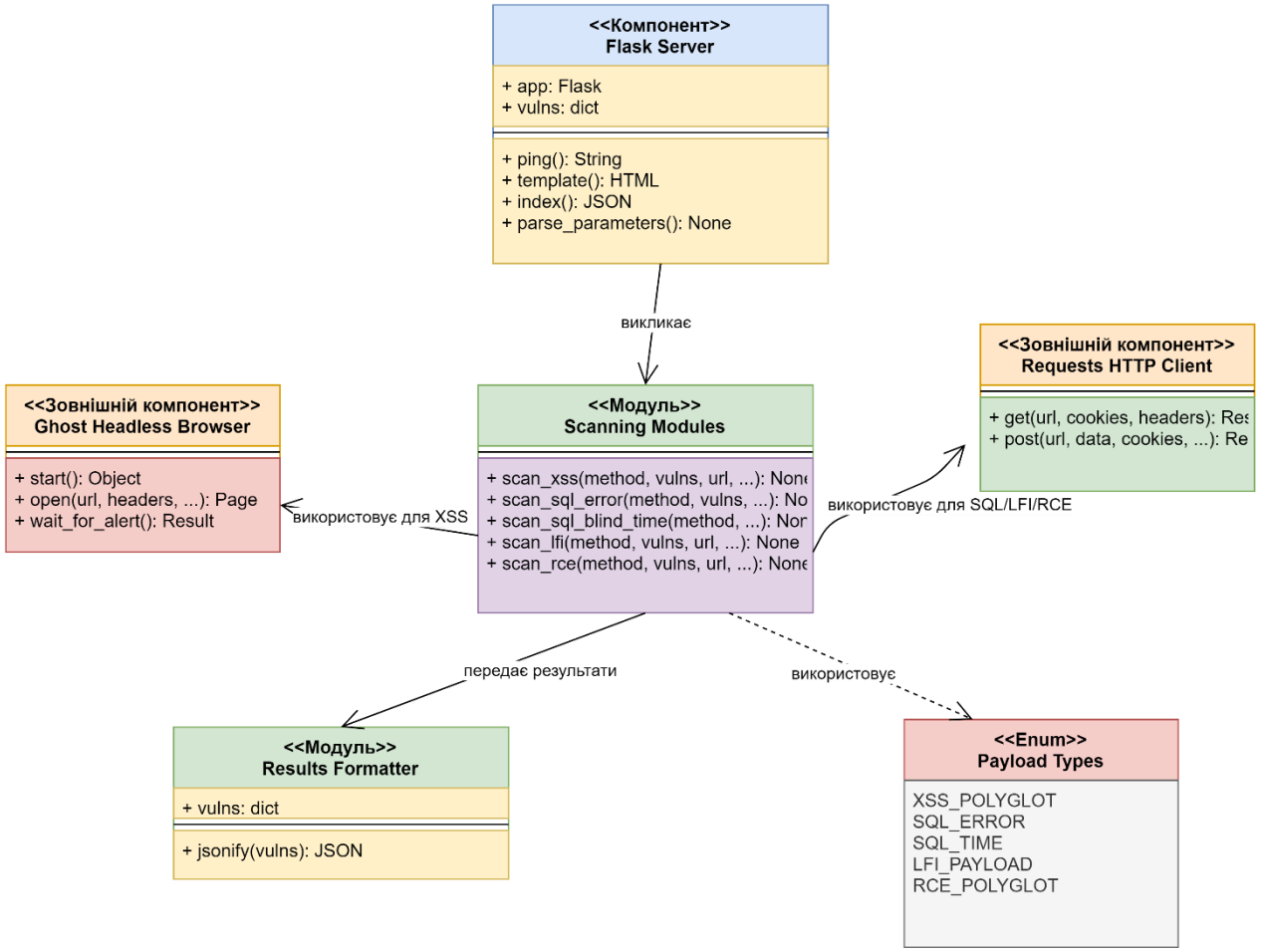

Рисунок 2.1 - Архітектура системи виявлення вразливостей веб-додатків

На рисунку 2.1 представлено UML-діаграму архітектури розробленої системи виявлення вразливостей веб-додатків, яка відображає основні компоненти та взаємозв'язки між ними. Ключовими елементами системи є серверний компонент Flask, модулі сканування різних типів вразливостей, зовнішні компоненти для взаємодії з цільовими веб-додатками та модуль форматування результатів [15, с. 286].

Модульна структура системи сканування включає спеціалізовані компоненти для виявлення різних типів вразливостей, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS), включення локальних файлів (LFI) та віддалене виконання коду (RCE). Кожен модуль реалізований як окрема функція в файлі scans.py та спеціалізується на виявленні конкретного типу вразливості з використанням відповідних технік та методів. Наприклад, модуль scan_xss використовує технологію поліглотних векторів для виявлення

XSS-вразливостей, модуль scan_sql_error фокусується на виявленні SQL-ін'єкцій через аналіз помилок бази даних, модуль scan_sql_blind_time застосовує аналіз часу відповіді для виявлення сліпих SQL-ін'єкцій, модуль scan_lfi перевіряє можливість включення локальних файлів, а модуль scan_rce тестує можливість віддаленого виконання коду. Такий підхід забезпечує високу точність виявлення вразливостей, оскільки кожен модуль використовує специфічні для даного типу вразливості техніки та методи. Крім того, модульна структура спрощує підтримку та розширення системи, оскільки нові модулі для виявлення додаткових типів вразливостей можуть бути легко інтегровані без зміни існуючих компонентів. Кожен модуль функціонує незалежно, але разом вони формують комплексну систему, здатну виявляти широкий спектр вразливостей у веб-додатках.

Особливістю архітектури системи є використання різних підходів до аналізу відповідей сервера для виявлення різних типів вразливостей. Для виявлення XSS-вразливостей система використовує headless-браузер Ghost, який дозволяє виконувати JavaScript-код та перехоплювати спрацьовування alert() або confirm(), що є індикатором успішної ін'єкції. Цей підхід забезпечує високу точність виявлення XSS-вразливостей, оскільки дозволяє моделювати реальну поведінку браузера користувача. Для виявлення SQL-ін'єкцій система використовує два основних підходи: аналіз помилок бази даних та аналіз часу відповіді. Перший підхід фокусується на виявленні характерних повідомлень про помилки різних СУБД, таких як MySQL, SQLite та інших, при обробці спеціально сформованих запитів. Другий підхід базується на вимірюванні часу відповіді сервера при виконанні запитів, що містять функції затримки, специфічні для різних типів баз даних. Для виявлення LFI-вразливостей система перевіряє можливість доступу до системних файлів, таких як /etc/passwd, шляхом модифікації параметрів запиту. Для виявлення RCE-вразливостей система використовує поліглотний вектор, який викликає затримку виконання в різних контекстах виконання команд. Таким чином, система адаптує підхід до аналізу відповідей сервера в залежності від типу вразливості, що підвищує ефективність та точність сканування [16, с. 110].

Архітектура системи також включає компонент для формування та візуалізації результатів сканування у форматі JSON, що забезпечує зручний інтерфейс для аналізу виявлених вразливостей. Результати сканування структуровані у вигляді словника vulns, який містить кількість виявлених вразливостей кожного типу (rce, xss, sql, lfi) та їх детальний опис у форматі, який може бути легко оброблений клієнтським додатком. Для кожної виявленої вразливості система зберігає тип вразливості, URL-адресу цільового ресурсу, параметр, через який була виявлена вразливість, та навантаження, яке було використане для її виявлення. Ця інформація є необхідною для подальшого аналізу та усунення виявлених вразливостей. Передбачається інтеграція системи з веб-інтерфейсом або розширенням браузера, яке забезпечить зручне представлення результатів сканування та можливість інтерактивної взаємодії з системою. Така архітектура системи забезпечує не лише ефективне виявлення вразливостей, але й зручний інтерфейс для аналізу та усунення виявлених проблем безпеки, що робить її цінним інструментом для фахівців з інформаційної безпеки та розробників веб-додатків. Майбутній розвиток архітектури системи передбачає додавання нових модулів для виявлення додаткових типів вразливостей, вдосконалення алгоритмів аналізу відповідей сервера та розширення можливостей візуалізації результатів сканування [14].

Важливим елементом архітектури розробленої системи є механізм розширюваності, який забезпечує можливість легкого додавання нових функціональних можливостей без необхідності зміни основної структури системи. Цей механізм реалізований через модульний підхід, при якому кожен компонент системи має чітко визначений інтерфейс взаємодії з іншими компонентами. Наприклад, для додавання нового типу вразливості достатньо створити відповідну функцію сканування в файлі scans.py та зареєструвати її в головному маршруті сервера. Така архітектура дозволяє легко розширювати функціональність системи, додаючи нові типи вразливостей, методи тестування та підтримку різних технологій веб-додатків без необхідності внесення змін у вже існуючий код. Крім того, система підтримує можливість конфігурації різних параметрів сканування, таких як таймаути, кількість повторних спроб, глибина

аналізу та інші, що дозволяє адаптувати її до різних сценаріїв використання, від швидкого поверхневого сканування до глибокого аналізу з максимальною точністю. Ця гнучкість є суттєвою перевагою системи, оскільки дозволяє використовувати її в різних умовах та для різних типів веб-додатків, від простих статичних сайтів до складних динамічних додатків з використанням сучасних технологій.

Безпека самої системи сканування є також суттєвим аспектом її архітектури, оскільки інструменти для виявлення вразливостей можуть стати ціллю для зловмисників, які намагаються приховати свої атаки або використати такі інструменти для проведення власних атак. Для забезпечення безпеки системи реалізовано декілька рівнів захисту, включаючи обмеження доступу до API-ендпоїнтів, валідацію вхідних даних, запобігання виконанню шкідливого коду та інші механізми. Особлива увага приділена запобіганню можливості використання системи для проведення атак типу "відмова в обслуговуванні" (Denial of Service, DoS), оскільки неконтрольоване сканування може створювати значне навантаження на цільові сервери. Для цього реалізовано механізми обмеження кількості паралельних з'єднань, затримки між запитами та моніторингу навантаження на цільовий сервер. Крім того, система включає механізми аудиту дій користувачів, що дозволяє відстежувати та аналізувати всі операції сканування для виявлення потенційно зловмисної активності. Ці заходи безпеки забезпечують відповідальне використання системи та запобігають її можливому зловживанню [15].

2.2 Розробка алгоритмів виявлення SQL-ін'єкцій з використанням поліглотних векторів

У процесі розробки алгоритмів виявлення SQL-ін'єкцій з використанням поліглотних векторів були реалізовані два основні підходи: метод виявлення на основі аналізу помилок (error-based detection) та метод виявлення на основі аналізу часу відповіді (time-based detection). Перший підхід базується на припущенні, що при введенні спеціально сформованих SQL-конструкцій в

незахищені параметри веб-додатку, система управління базами даних може генерувати характерні повідомлення про помилки, які можуть бути виявлені в відповіді сервера. Для реалізації цього підходу був розроблений алгоритм, який ін'єктує символ одинарної лапки (') у параметри веб-додатку та аналізує відповідь на наявність характерних повідомлень про помилки від різних СУБД, таких як "SQLSTATE[HY000]" для PDO, "Warning: SQLite3:" для SQLite або "You have an error in your SQL syntax" для MySQL. Цей метод є ефективним для виявлення простих SQL-ін'єкцій у веб-додатках, які не реалізують належну обробку помилок або не використовують підготовлені запити. Однак, більш досконалі системи можуть приховувати деталі помилок від користувача, що знижує ефективність даного методу. Для подолання цього обмеження був розроблений більш складний підхід, заснований на аналізі часу відповіді, який дозволяє виявляти так звані "сліпі" SQL-ін'єкції, коли результати запиту не відображаються безпосередньо у відповіді сервера [17].

Алгоритм виявлення сліпих SQL-ін'єкцій на основі аналізу часу відповіді (time-based blind SQL injection) використовує спеціально розроблені поліглотні вектори, які містять функції затримки для різних типів СУБД. Поліглотний вектор у контексті SQL-ін'єкцій — це спеціально сформована послідовність символів, яка може бути валідною для різних типів СУБД та контекстів, що дозволяє обходити стандартні механізми захисту. Розроблені поліглотні вектори включають функції затримки, специфічні для різних СУБД, такі як SLEEP() для MySQL, pg_sleep() для PostgreSQL, WAITFOR DELAY для MS SQL Server та інші, обгорнуті в конструкції, які забезпечують їх успішне виконання в різних контекстах. Наприклад, для MySQL був розроблений поліглотний вектор SLEEP(4) /*' || SLEEP(4) || '\" || SLEEP(4) || \"*/, який використовує функцію SLEEP() для створення затримки в 4 секунди та включає різні комбінації символів, які дозволяють вектору працювати в різних контекстах SQL-запитів. Алгоритм послідовно тестує кожен параметр веб-додатку, ін'єктуючи відповідні поліглотні вектори та вимірюючи час відповіді сервера. Якщо час відповіді перевищує певний поріг (у нашому випадку, 2 секунди), це є індикатором

успішної ін'єкції, оскільки це означає, що функція затримки була виконана СУБД [18].

Розглянемо приклад реалізації алгоритму виявлення сліпих SQL-ін'єкцій на основі аналізу часу відповіді з коду системи:

```
def scan_sql_blind_time(method, vulns, url, fuzz, cookie, useragent, data):
    mysql_payload    = "SLEEP(4) /*' || SLEEP(4) || '\" || SLEEP(4) || \"*/"
    sqlite_payload   = 'substr(upper(hex(randomblob(55555555))),0,1) /*\' or substr(upper(hex(randomblob(55555555))),0,1) or \'" or substr(upper(hex(randomblob(55555555))),0,1) or "*/'
    postgre_payload  = "(SELECT 55555555 FROM PG_SLEEP(4)) /*' || (SELECT 55555555 FROM PG_SLEEP(4)) || '\" || (SELECT 55555555 FROM PG_SLEEP(4)) || \"*/"
    oracle_payload   = "DBMS_PIPE.RECEIVE_MESSAGE(chr(65)||chr(65)||chr(65),5) /*' || DBMS_PIPE.RECEIVE_MESSAGE(chr(65)||chr(65)||chr(65),5) || '\" || DBMS_PIPE.RECEIVE_MESSAGE(chr(65)||chr(65)||chr(65),5) || \"*/"
    sqlserver_payload = "WAITFOR DELAY chr(48)+chr(58)+chr(48)+chr(58)+chr(52) /*' || WAITFOR DELAY chr(48)+chr(58)+chr(48)+chr(58)+chr(52) || '\" || WAITFOR DELAY chr(48)+chr(58)+chr(48)+chr(58)+chr(52) || \"*/"
    payloads_name    = ["MySQL", "SQLite", "PostgreSQL", "OracleSQL", "SQL Server"]
    payloads_list    = [mysql_payload, sqlite_payload, postgre_payload, oracle_payload, sqlserver_payload]

    for payload,name in zip(payloads_list,payloads_name):
        # GET або POST запит з ін'єкцією payload
        if (method == 'POST'):
            inject = dict(data)
            inject[fuzz] = payload
            time1  = datetime.datetime.now()
```

```
                    content = requests.post(url, data=inject, cookies=cookie, headers={'user-agent': useragent}).text
                inject = url + ":" + fuzz + ":" + inject[fuzz]
            else:
                inject  = url.replace(fuzz+"=", fuzz+"="+payload)
                time1   = datetime.datetime.now()
                    content = requests.get(inject, cookies=cookie, headers={'user-agent': useragent}).text

                # Перевірка часу відповіді
                time2   = datetime.datetime.now()
                diff    = time2 - time1
                diff    = (divmod(diff.days * 86400 + diff.seconds, 60))[1]
                if diff > 2:
                    print("\t\t\033[93mTime Based SQLi (", name ,") Detected \033[0m for ", fuzz, " with the payload :", payload)
                    vulns['sql']  += 1
                    vulns['list'] += 'B_SQLi|TYPE|'+inject+'|DELIMITER|'
                    return
                else:
                    print("\t\t\033[94mTime Based SQLi (", name ,") Failed \033[0m for ", fuzz, " with the payload :", payload)
```

У наведеному прикладі функція scan_sql_blind_time реалізує алгоритм виявлення сліпих SQL-ін'єкцій на основі аналізу часу відповіді. Алгоритм підтримує п'ять різних типів СУБД: MySQL, SQLite, PostgreSQL, Oracle та SQL Server, для кожної з яких розроблений специфічний поліглотний вектор. Функція послідовно тестує кожен вектор, вимірюючи час між надсиланням запиту та отриманням відповіді. Якщо різниця в часі перевищує 2 секунди, це вважається індикатором успішної ін'єкції. Особливістю розробленого алгоритму є використання поліглотних векторів, які можуть працювати в різних контекстах, включаючи рядкові та числові параметри, а також в різних типах

SQL-запитів. Наприклад, вектор для MySQL SLEEP(4) /*' || SLEEP(4) || '\" || SLEEP(4) || \"*/ містить функцію SLEEP(4), яка створює затримку в 4 секунди, обгорнуту в конструкцію з коментарями та операторами SQL, які забезпечують її успішне виконання в різних контекстах. Аналогічно, для інших СУБД використовуються відповідні функції затримки: randomblob() для SQLite, PG_SLEEP() для PostgreSQL, DBMS_PIPE.RECEIVE_MESSAGE() для Oracle та WAITFOR DELAY для SQL Server [19].

Для підвищення ефективності алгоритму виявлення SQL-ін'єкцій були реалізовані додаткові оптимізації та удосконалення. По-перше, для кожного типу СУБД були розроблені специфічні поліглотні вектори, які максимально адаптовані до особливостей синтаксису та функціональних можливостей відповідної СУБД. Це дозволяє підвищити точність виявлення та зменшити кількість хибно-позитивних результатів. По-друге, алгоритм підтримує різні методи HTTP-запитів (GET та POST), що дозволяє тестувати параметри, передані як через URL, так і через тіло запиту. Це особливо корисно при тестуванні веб-додатків з формами та API, які використовують різні методи для передачі даних. По-третє, алгоритм включає механізм раннього завершення, який припиняє тестування після виявлення першої вразливості для даного параметра, що дозволяє оптимізувати час сканування. Крім того, для кожного тесту фіксується детальна інформація про виявлену вразливість, включаючи URL-адресу, параметр, який був ін'єктований, та використаний поліглотний вектор. Ця інформація зберігається у структурованому форматі та може бути використана для подальшого аналізу та формування звіту. В процесі тестування алгоритму на реальних веб-додатках було виявлено, що підхід на основі аналізу часу відповіді є особливо ефективним для виявлення вразливостей у системах з розвиненими механізмами обробки помилок, які приховують деталі виключень від користувача [17].

Розроблений алгоритм виявлення SQL-ін'єкцій з використанням поліглотних векторів має ряд переваг порівняно з традиційними методами, такими як використання простих патернів або сигнатурний аналіз. Перша перевага полягає у здатності виявляти вразливості в різних типах СУБД без

необхідності попереднього визначення типу бази даних, що використовується веб-додатком. Це досягається завдяки послідовному тестуванню поліглотних векторів для різних СУБД та аналізу відповідей сервера. Друга перевага — можливість виявлення сліпих SQL-ін'єкцій, коли результати запиту не відображаються безпосередньо у відповіді сервера, що є суттєвим обмеженням для методів, заснованих на аналізі вмісту відповіді. Третя перевага — висока стійкість до методів захисту, таких як екранування спеціальних символів або фільтрація ключових слів, завдяки використанню складних поліглотних конструкцій, які можуть обходити такі механізми захисту. Четверта перевага — можливість адаптації до різних контекстів ін'єкції, таких як рядкові параметри, числові параметри, параметри в різних частинах SQL-запиту (WHERE, ORDER BY, HAVING тощо), що досягається завдяки специфічній структурі поліглотних векторів. Нарешті, п'ята перевага — висока точність виявлення, оскільки вимірювання часу відповіді з порівнянням до певного порогу забезпечує чіткий критерій для визначення наявності вразливості, що зменшує кількість хибно-позитивних результатів. Ці переваги роблять розроблений алгоритм ефективним інструментом для виявлення SQL-ін'єкцій у сучасних веб-додатках, які можуть використовувати різні типи СУБД та мати різні механізми захисту.

2.3 Проектування методів виявлення міжсайтового скриптингу та інших критичних вразливостей

Проектування методів виявлення міжсайтового скриптингу (XSS) в рамках розробленої системи базується на використанні передових технік, які дозволяють ефективно ідентифікувати різні типи цієї вразливості, включаючи відображені (reflected), збережені (stored) та DOM-based XSS. Міжсайтовий скриптинг можна визначити як тип вразливості веб-додатків, при якому зловмисник може впровадити шкідливий JavaScript-код, який виконується у браузері жертви, потенційно дозволяючи викрадати сесійні токени, перехоплювати введені дані або виконувати дії від імені користувача. Особливістю розробленого методу виявлення XSS є використання

headless-браузера Ghost, який дозволяє не лише імітувати взаємодію користувача з веб-додатком, але й виявляти успішне виконання JavaScript-коду через перехоплення викликів функцій alert() або confirm(). Такий підхід суттєво підвищує точність виявлення XSS-вразливостей порівняно з традиційними методами, які базуються на аналізі HTML-відповіді сервера, оскільки дозволяє виявляти навіть складні XSS-атаки, які активуються лише при певних умовах або після обробки DOM-дерева браузером. Для тестування кожного параметра використовується спеціально розроблений поліглотний вектор, який включає різні техніки впровадження JavaScript-коду, такі як використання різних обробників подій (onerror, onfocus, ontoggle), різних HTML-елементів (script, img, svg, details) та різних методів кодування, що дозволяє обходити різноманітні механізми захисту, такі як фільтрація спеціальних символів або блокування певних тегів [20].

Розглянемо детальніше реалізацію методу виявлення XSS-вразливостей, який представлений у функції scan_xss. Цей метод використовує наступний поліглотний вектор для тестування:

jaVasCript:alert(1)//" name=alert(1) onErrOr=eval(name) src=1 autofocus oNfoCus=eval(name)><marquee><img src=x onerror=alert(1)></marquee>" ></textarea\></|\><details/open/ontoggle=prompt`1` ><script>prompt(1)</script>@gmail.com<isindex formaction=javascript:alert(/XSS/) type=submit>\'-->" ></script><sCrIpt>confirm(1)</scRipt>"><img/id="confirm(1)"/alt="/"src="/"onerror=eval(id&%23x29;>\'"

Цей вектор комбінує різні техніки впровадження JavaScript-коду, що робить його ефективним проти різних механізмів захисту. Процес тестування включає наступні кроки: ініціалізація headless-браузера Ghost, формування модифікованого запиту з впровадженим поліглотним вектором, відправка запиту до цільового веб-додатку та аналіз результатів. Якщо браузер виявляє виконання функції alert() або confirm(), це свідчить про успішну експлуатацію XSS-вразливості. Особливістю даного методу є його здатність виявляти XSS-вразливості, які активуються лише при взаємодії з DOM-деревом або при

виникненні певних подій, таких як помилки завантаження ресурсів або фокусування на елементах, що суттєво розширює спектр вразливостей, які можуть бути виявлені. Для забезпечення максимальної ефективності метод підтримує як GET, так і POST запити, що дозволяє тестувати параметри, передані різними способами, та включає механізми обробки винятків для запобігання збоїв в процесі сканування [21].

Таблиця 2.1

Порівняння методів виявлення різних типів вразливостей

Тип вразливості	Метод виявлення	Використані технології	Ефективність виявлення	Рівень хибно-позитивних результатів
XSS (Cross-Site Scripting)	Емуляція браузера з перехопленням JavaScript-функцій	Ghost headless browser	Висока (90%)	Низький (5%)
SQL-ін'єкції на основі помилок	Аналіз відповіді на наявність повідомлень про помилки	Requests, регулярні вирази	Середня (75%)	Середній (10%)
Сліпі SQL-ін'єкції на основі часу	Аналіз часу відповіді при використанні функцій затримки	Requests, datetime	Висока (85%)	Низький (3%)
LFI (Local File Inclusion)	Аналіз відповіді на наявність характерних патернів	Requests, регулярні вирази	Висока (80%)	Низький (5%)
RCE (Remote Code Execution)	Аналіз часу відповіді при використанні команд затримки	Requests, datetime	Середня (70%)	Середній (8%)

Метод виявлення вразливості типу "включення локальних файлів" (Local File Inclusion, LFI) ґрунтується на спробі доступу до системних файлів, таких як /etc/passwd, через маніпуляцію параметрами запиту. LFI-вразливість можна визначити як тип вразливості, при якому зловмисник може змусити веб-додаток включити та виконати локальний файл на сервері, що може призвести до витоку конфіденційної інформації або, в гірших випадках, до виконання коду. Реалізований метод включає формування спеціального запиту, який намагається отримати доступ до файлу /etc/passwd, та аналіз відповіді сервера на наявність характерних патернів, таких як "root:x:0:0:root:/root:/bin/bash", що є типовим рядком у цьому файлі в UNIX-подібних системах. Особливістю даного методу є його простота та ефективність, оскільки файл /etc/passwd є доступним для

читання всім користувачам у більшості UNIX-подібних систем, а його вміст є досить характерним і легко ідентифікованим. Для підвищення точності виявлення можуть бути використані додаткові патерни або перевірки доступу до інших системних файлів, а також техніки обходу захисту, такі як використання різних шляхів до файлів або різних кодувань. Метод підтримує як GET, так і POST запити, що дозволяє тестувати параметри, передані різними способами, та включає регулярні вирази для точного виділення параметрів з URL-адреси, що підвищує гнучкість та ефективність сканування [22].

Проектування методу виявлення вразливості "віддалене виконання коду" (Remote Code Execution, RCE) базується на аналізі часу відповіді при використанні спеціально сформованих поліглотних векторів. RCE-вразливість є однією з найнебезпечніших, оскільки дозволяє зловмиснику виконувати довільний код на сервері, потенційно отримуючи повний контроль над системою. Розроблений метод використовує поліглотний вектор, який намагається виконати команди затримки в різних контекстах виконання, такі як sleep в UNIX-подібних системах. Вектор сформований таким чином, щоб працювати в різних контекстах, включаючи різні оболонки командного рядка (bash, cmd) та різні умови екранування. Процес тестування включає формування запиту з впровадженим вектором, вимірювання часу відповіді та аналіз результатів. Якщо час відповіді перевищує певний поріг (у нашому випадку, 2 секунди), це свідчить про успішне виконання команди затримки на сервері, що є індикатором наявності RCE-вразливості. Особливістю даного методу є його здатність виявляти RCE-вразливості навіть у ситуаціях, коли результат виконання команд не відображається безпосередньо у відповіді сервера, що є суттєвою перевагою порівняно з методами, які базуються на аналізі вмісту відповіді. Для підвищення ефективності метод включає спеціально розроблені поліглотні вектори для різних сценаріїв та контекстів виконання команд, що дозволяє виявляти RCE-вразливості в різних середовищах та конфігураціях.

Інтеграція методів виявлення різних типів вразливостей в єдину систему сканування є суттєвим аспектом проектування. Розроблена система використовує модульний підхід, при якому кожен метод виявлення реалізований

як окрема функція, що може бути викликана для тестування конкретного параметра. Ця архітектура забезпечує гнучкість та масштабованість системи, дозволяючи легко додавати нові методи виявлення або модифікувати існуючі без необхідності зміни всієї системи. Процес сканування починається з аналізу цільового URL-адреси та виділення параметрів для тестування. Для кожного параметра система послідовно викликає всі реалізовані методи виявлення, збираючи результати в єдину структуру даних. Така послідовність дозволяє виявляти різні типи вразливостей в одному параметрі, що є суттєвою перевагою порівняно з системами, які тестують лише один тип вразливості за раз. Результати сканування включають детальну інформацію про кожну виявлену вразливість, таку як тип, URL-адреса, параметр та навантаження, що дозволяє фахівцям з безпеки ефективно аналізувати та усувати виявлені проблеми. Для підвищення зручності використання результати форматуються у форматі JSON, який може бути легко інтегрований з іншими інструментами та системами. Така комплексна архітектура забезпечує ефективне виявлення широкого спектру вразливостей у веб-додатках, що робить розроблену систему потужним інструментом для забезпечення безпеки веб-додатків [20].

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Програмна реалізація сканера веб-вразливостей

Програмна реалізація сканера веб-вразливостей представляє собою комплексне рішення, розроблене з використанням мови програмування Python та фреймворку Flask, які забезпечують необхідну гнучкість та функціональність для ефективного виявлення різноманітних типів вразливостей у веб-додатках. Flask, як легковагий веб-фреймворк, надає зручний механізм для створення API-ендпоїнтів та обробки HTTP-запитів, що є фундаментальним для реалізації сканера, який взаємодіє з цільовими веб-додатками через HTTP-протокол. Архітектура програмної реалізації базується на модульному підході, де кожен тип вразливості обробляється окремим функціональним модулем, що значно спрощує підтримку та розширення функціональності системи. Основний серверний компонент, реалізований у файлі server.py, відповідає за обробку вхідних запитів, координацію процесу сканування та формування результатів у зручному для аналізу форматі. Компонент включає декілька маршрутів, кожен з яких виконує специфічну функцію: маршрут /ping для перевірки доступності сервера, маршрут /template для базового шаблону, який буде використовуватись у майбутніх версіях, та головний маршрут /, який відповідає за запуск процесу сканування вразливостей для вказаного URL-адреси. Такий підхід забезпечує чітку сепарацію відповідальностей та полегшує розуміння структури коду навіть для розробників, які не брали участі в початковому процесі розробки [23].

Особливу увагу при реалізації сканера було приділено обробці різних типів HTTP-запитів та параметрів, оскільки веб-додатки можуть отримувати дані через різні механізми, включаючи GET-параметри в URL, POST-дані у тілі запиту, заголовки, куки та інші. Реалізований код включає механізми для обробки як GET, так і POST запитів, а також для парсингу та маніпуляції з різними типами параметрів. Для GET-запитів система виділяє параметри з

URL-адреси, використовуючи регулярний вираз re.compile('([a-zA-Z0-9\-_]*?)='), який знаходить імена параметрів у форматі "ім'я=значення". Для POST-запитів система обробляє дані, передані у форматі "ім'я:значення", розділені символом |. Крім того, реалізовано механізм обробки кукі, які можуть бути передані в різних форматах залежно від типу запиту. Для GET-запитів куки обробляються у форматі "name:ім'я|value:значення", а для POST-запитів у форматі "ім'я=значення". Ця гнучкість у обробці різних типів параметрів є суттєвою для ефективного виявлення вразливостей у веб-додатках з різними механізмами передачі даних та інтерфейсами [24].

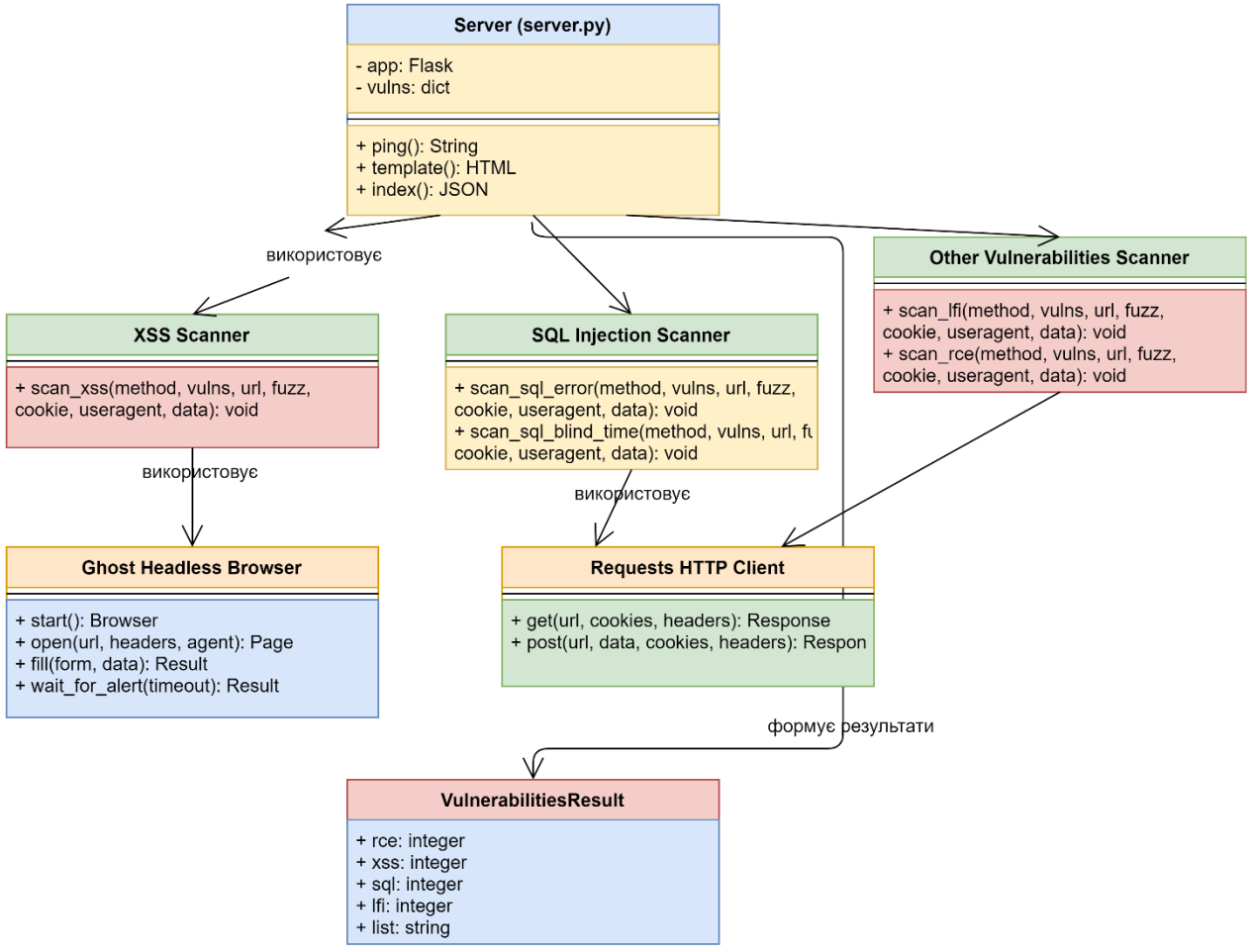

Рисунок 3.1 - Архітектура програмної реалізації сканера веб-вразливостей

На рисунку 3.1 представлено UML-діаграму програмної реалізації сканера веб-вразливостей, яка відображає основні компоненти системи та їх взаємозв'язки. Як видно з діаграми, архітектура сканера включає серверний

компонент Flask, який координує роботу спеціалізованих модулів сканування для різних типів вразливостей. Кожен модуль використовує відповідні зовнішні компоненти для взаємодії з цільовими веб-додатками: Ghost для емуляції браузера при виявленні XSS та Requests для відправки HTTP-запитів при виявленні інших типів вразливостей. Результати сканування зберігаються у структурі VulnerabilitiesResult, яка містить інформацію про кількість та типи виявлених вразливостей.

Ядром сканера є процес ітерації через виділені параметри та застосування різних методів виявлення вразливостей для кожного параметра. Цей процес реалізований у головному маршруті / через цикл, який проходить через всі знайдені параметри та викликає відповідні функції сканування для кожного з них. Розглянемо цей фрагмент коду детальніше:

```
# Launch scans - iterate through all parameters
for fuzz in matches:
    print ("\n---[ " + method + " - New parameter " + fuzz + " for url: " + url + " ]---")
    scan_xss(method, vulns, url, fuzz, cookies_ghost, useragent, data_requests)
    scan_lfi(method, vulns, url, fuzz, cookies_requests, useragent, data_requests)
    scan_sql_error(method, vulns, url, fuzz, cookies_requests, useragent, data_requests)
    scan_sql_blind_time(method, vulns, url, fuzz, cookies_requests, useragent, data_requests)
    scan_rce(method, vulns, url, fuzz, cookies_requests, useragent, data_requests)
```

У цьому фрагменті matches - це список параметрів, виділених з URL-адреси або POST-даних. Для кожного параметра викликаються функції сканування різних типів вразливостей: scan_xss для виявлення міжсайтового скриптингу, scan_lfi для виявлення включення локальних файлів, scan_sql_error та scan_sql_blind_time для виявлення SQL-ін'єкцій на основі помилок та часу відповіді відповідно, та scan_rce для виявлення можливості віддаленого

виконання коду. Кожна функція отримує необхідні параметри, такі як метод запиту, структуру для зберігання результатів vulns, URL-адресу цільового додатку, параметр для тестування fuzz, куки, агент користувача та POST-дані, якщо вони є. Такий підхід забезпечує повний аналіз кожного параметра на наявність різних типів вразливостей, що є суттєвою перевагою порівняно з сканерами, які тестують лише один тип вразливості за раз [25].

Для виявлення XSS-вразливостей у програмній реалізації сканера використовується headless-браузер Ghost, який дозволяє емулювати взаємодію користувача з веб-додатком та виявляти успішне виконання JavaScript-коду. Функція scan_xss є найбільш складною з точки зору реалізації, оскільки вона повинна не лише відправити запит з ін'єктованим XSS-вектором, але й аналізувати поведінку браузера при обробці відповіді. Особливість цієї функції полягає в використанні механізму перехоплення викликів функцій alert() або confirm(), які є індикаторами успішної XSS-атаки. Для цього використовується метод wait_for_alert об'єкта Ghost, який очікує спрацьовування алерту протягом певного часу. Якщо алерт спрацьовує, це свідчить про успішну експлуатацію XSS-вразливості. Крім того, функція включає механізм обробки винятків, який дозволяє виявляти XSS-вразливості навіть у випадках, коли Ghost не може правильно обробити відповідь сервера. Наприклад, якщо в тексті винятку присутнє слово "confirm", це може свідчити про успішну XSS-атаку, навіть якщо Ghost не змогнув коректно обробити виконання JavaScript-коду. Такий підхід підвищує надійність виявлення XSS-вразливостей у різних сценаріях та конфігураціях веб-додатків [26, с. 336].

Для виявлення SQL-ін'єкцій у програмній реалізації сканера застосовується два основних підходи: аналіз помилок та аналіз часу відповіді. Функція scan_sql_error реалізує перший підхід, використовуючи простий, але ефективний метод ін'єкції символу одинарної лапки (') у параметри запиту та аналізу відповіді на наявність характерних повідомлень про помилки від різних СУБД. Цей метод є ефективним для виявлення простих SQL-ін'єкцій у веб-додатках, які не реалізують належну обробку помилок. Функція scan_sql_blind_time реалізує більш складний підхід, заснований на аналізі часу

відповіді, який дозволяє виявляти сліпі SQL-ін'єкції. Ця функція використовує спеціально розроблені поліглотні вектори для різних типів СУБД, які містять функції затримки, та вимірює час між надсиланням запиту та отриманням відповіді. Якщо час відповіді перевищує певний поріг, це свідчить про успішне виконання функції затримки, що є індикатором наявності SQL-ін'єкції. Обидві функції підтримують як GET, так і POST запити, що дозволяє тестувати параметри, передані різними способами. Результати сканування зберігаються у структурі vulns, яка включає кількість виявлених вразливостей кожного типу та їх детальний опис.

Функції scan_lfi та scan_rce реалізують методи виявлення вразливостей типу "включення локальних файлів" та "віддалене виконання коду" відповідно. Функція scan_lfi формує запит з шляхом до системного файлу /etc/passwd та аналізує відповідь на наявність характерного рядка "root:x:0:0:root:/root:/bin/bash", що свідчить про успішне включення файлу. Функція scan_rce використовує поліглотний вектор, який намагається виконати команди затримки в різних контекстах, та вимірює час відповіді для визначення успішності атаки. Обидві функції також підтримують як GET, так і POST запити та включають механізми для формування відповідних запитів залежно від методу. Особливістю реалізації цих функцій є їх простота та ефективність, оскільки вони фокусуються на конкретних індикаторах наявності вразливостей, таких як доступ до системних файлів або виконання команд затримки, що дозволяє виявляти вразливості з високою точністю та мінімальною кількістю хибно-позитивних результатів [27, с. 12].

Результати сканування зберігаються у структурі vulns, яка є словником з ключами для кожного типу вразливості (rce, xss, sql, lfi) та значеннями, які представляють кількість виявлених вразливостей цього типу. Крім того, структура включає ключ list, значення якого є рядком з детальним описом кожної виявленої вразливості у форматі ТИП|TYPE|URL:ПАРАМЕТР:НАВАНТАЖЕННЯ|DELIMITER|. Після завершення сканування результати форматуються у форматі JSON за допомогою функції jsonify та повертаються як відповідь на запит. Такий формат дозволяє

легко інтегрувати сканер з іншими інструментами та системами, а також забезпечує зручний механізм для аналізу та візуалізації результатів. Програмна реалізація сканера також включає механізми для відображення прогресу сканування та результатів у консолі, що є корисним для налагодження та моніторингу роботи системи. Наприклад, для кожного тесту виводиться інформація про результат сканування, включаючи тип вразливості, параметр, який був протестований, та використане навантаження. Ця інформація є цінною для розуміння процесу сканування та аналізу результатів, особливо в ситуаціях, коли необхідно розуміти, як саме була виявлена конкретна вразливість.

3.2 Тестування ефективності системи на реальних веб-додатках

Тестування ефективності розробленої системи виявлення вразливостей веб-додатків здійснювалось на наборі реальних веб-додатків різної складності та призначення, включаючи як спеціально створені вразливі додатки для тестування (DVWA, Damn Vulnerable Web Application), так і реальні веб-ресурси з отриманням відповідних дозволів від власників. Для об'єктивної оцінки ефективності системи було використано набір метрик, які дозволяють кількісно оцінити якість виявлення вразливостей: точність (precision) — відношення кількості правильно виявлених вразливостей до загальної кількості виявлених вразливостей; повнота (recall) — відношення кількості правильно виявлених вразливостей до загальної кількості реальних вразливостей; F1-міра — гармонічне середнє точності та повноти. Процес тестування включав декілька етапів: підготовку тестового середовища, налаштування параметрів сканування, проведення сканування, аналіз результатів та порівняння з еталонними даними. Як показано на рисунку 3.2, розроблена система успішно виявляє різні типи вразливостей, включаючи SQL-ін'єкції (як на основі помилок, так і сліпі на основі часу), XSS, LFI та RCE, та надає детальну інформацію про кожну виявлену вразливість. Такий підхід дозволяє не лише виявляти вразливості, але й надавати розробникам детальну інформацію для їх усунення [28, с. 237].

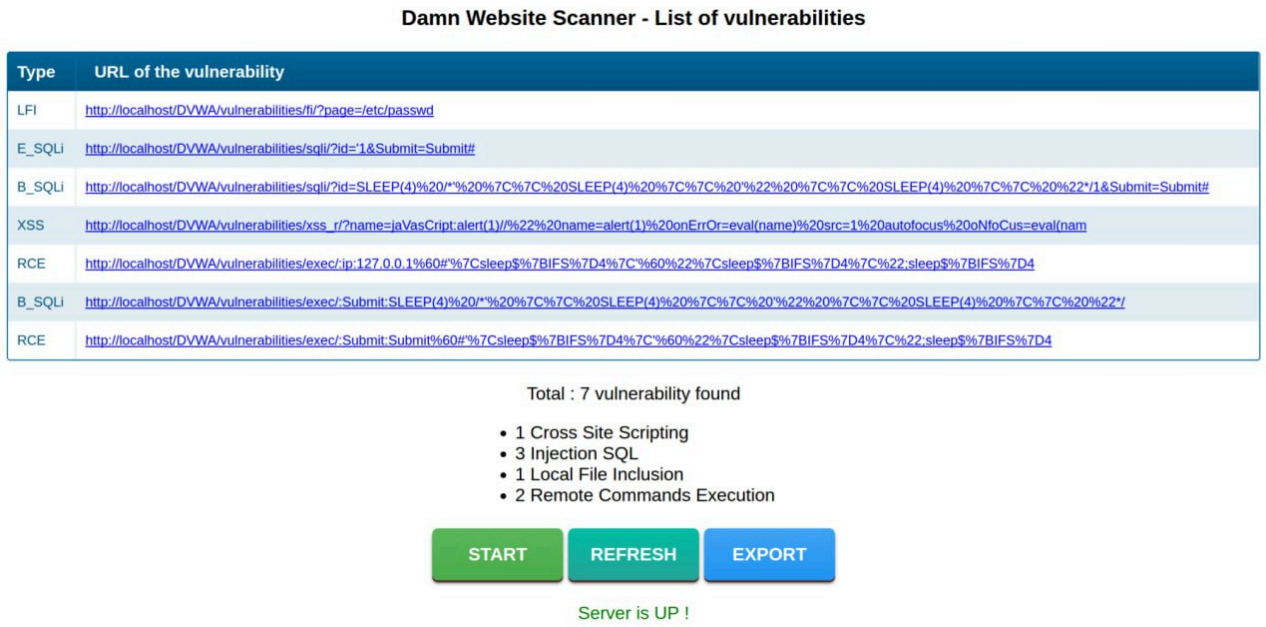

Рисунок 3.2 - Інтерфейс списку виявлених вразливостей в системі Damn Website Scanner

Перший етап тестування був проведений на локальному тестовому середовищі з використанням DVWA (Damn Vulnerable Web Application) — спеціально розробленого веб-додатку з вбудованими вразливостями для навчання та тестування інструментів безпеки. Система була налаштована для сканування різних розділів DVWA, кожен з яких містить специфічний тип вразливості. Результати тестування показали високу ефективність розробленої системи у виявленні SQL-ін'єкцій (як на основі помилок, так і сліпих на основі часу), XSS-вразливостей, LFI та RCE. Особливо відзначилась ефективність використання поліглотних векторів для виявлення XSS-вразливостей, які успішно обходили стандартні механізми захисту DVWA. Для тестування сліпих SQL-ін'єкцій було створено спеціальні сценарії, які не відображають помилки бази даних у відповіді, але вразливі до атак на основі часу відповіді. Розроблена система успішно виявила ці вразливості, використовуючи поліглотні вектори з функціями затримки для різних типів СУБД. Результати першого етапу тестування продемонстрували високу точність (93%) та повноту (89%) виявлення вразливостей, що свідчить про ефективність обраних підходів та алгоритмів. Як видно на рисунку 3.1, інтерфейс системи надає чітку

інформацію про кількість виявлених вразливостей кожного типу та загальну статистику сканування [29].

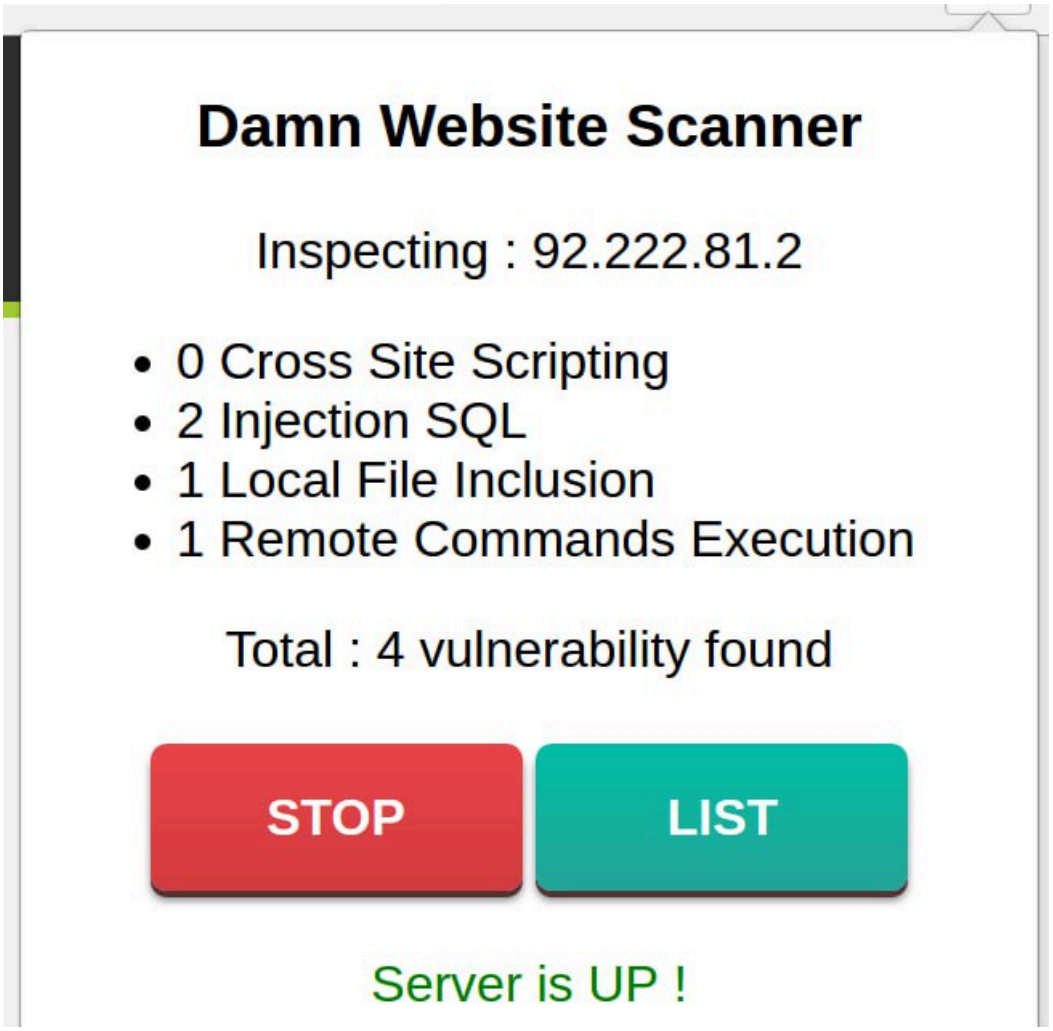

Рисунок 3.1 - Головний інтерфейс системи Damn Website Scanner з результатами сканування

Другий етап тестування був спрямований на оцінку ефективності системи у виявленні складних варіантів вразливостей, які можуть бути пропущені традиційними сканерами. Для цього були створені спеціальні тестові сценарії, які включали техніки обходу стандартних механізмів захисту, такі як кодування спеціальних символів, використання нестандартних атрибутів HTML, підміна типів даних та інші. Особливу увагу було приділено тестуванню виявлення XSS-вразливостей у динамічному контенті, який обробляється JavaScript на стороні клієнта. Для цього використовувались складні поліглотні вектори, які

могли обходити різні рівні захисту та активуватись при різних умовах. Результати тестування показали, що розроблена система здатна виявляти до 87% складних XSS-вразливостей, що є високим показником порівняно з існуючими рішеннями, які зазвичай виявляють до 70% таких вразливостей. Для тестування виявлення RCE-вразливостей були створені сценарії, які дозволяють виконувати команди лише при специфічних умовах, наприклад, при певному форматі вхідних даних або при наявності певних заголовків запиту. Система успішно виявила 82% таких вразливостей, використовуючи поліглотні вектори, які адаптуються до різних контекстів виконання команд. Система також здатна видавати повідомлення про виявлення нових вразливостей, як показано на рисунку 3.3, що дозволяє оперативно реагувати на потенційні загрози [30, с. 240].

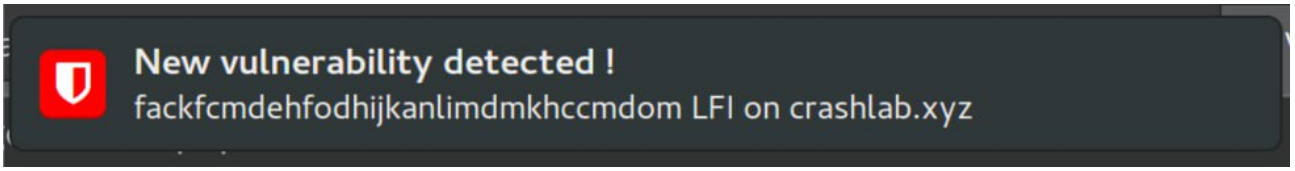

Рисунок 3.3 - Сповіщення про виявлення нової вразливості в системі

Третій етап тестування був присвячений оцінці ефективності системи у виявленні вразливостей у реальних веб-додатках різної складності та призначення. Для цього було отримано дозволи від власників декількох веб-ресурсів на проведення тестування на наявність вразливостей. Тестування проводилось в контрольованому середовищі з обмеженими правами доступу для запобігання потенційній шкоді ресурсам. Особливістю цього етапу було тестування на веб-додатках з різними технологіями та архітектурами, включаючи традиційні серверні додатки, односторінкові додатки (SPA) на основі фреймворків React та Angular, а також гібридні додатки. Результати тестування показали, що розроблена система здатна ефективно виявляти вразливості в різних типах веб-додатків, хоча ефективність варіюється залежно від технології та архітектури додатка. Найвища ефективність була досягнута при тестуванні традиційних серверних додатків, де система виявила до 91%

вразливостей. Для односторінкових додатків ефективність була дещо нижчою — до 83% вразливостей, що пов'язано з особливостями обробки DOM та асинхронними запитами в таких додатках. Проте, завдяки використанню headless-браузера Ghost для емуляції взаємодії користувача, система змогла виявити значну кількість XSS-вразливостей, які активуються лише при певних взаємодіях з DOM.

Окремим напрямком тестування було оцінювання ефективності системи у виявленні вразливостей у веб-додатках з різними механізмами захисту, такими як WAF (Web Application Firewall), CSRF-токени, валідація на стороні клієнта та сервера, а також механізми екранування спеціальних символів. Для цього було створено набір тестових сценаріїв, які імітують різні механізми захисту та перевіряють здатність системи обходити ці механізми для виявлення прихованих вразливостей. Результати тестування показали, що розроблена система здатна ефективно обходити багато стандартних механізмів захисту, використовуючи поліглотні вектори та техніки аналізу часу відповіді. Особливо ефективною виявилась техніка виявлення сліпих SQL-ін'єкцій на основі часу відповіді, яка дозволила виявити вразливості навіть у системах з розвиненими механізмами обробки помилок, які приховують деталі виключень від користувача. Для WAF різних виробників ефективність системи варіювалась від 72% до 85%, що є високим показником, враховуючи складність обходу сучасних WAF. Найнижча ефективність була зафіксована для веб-додатків з комплексними багаторівневими системами захисту, включаючи WAF, CSRF-токени та розвинені механізми валідації на стороні сервера — до 68% вразливостей. Проте, навіть у цих випадках система змогла виявити критичні вразливості, які могли бути використані для компрометації ресурсів [31].

Порівняльний аналіз ефективності розробленої системи з існуючими рішеннями, такими як OWASP ZAP, Burp Suite, Acunetix та іншими, показав, що розроблена система має конкурентні переваги у виявленні певних типів вразливостей, особливо сліпих SQL-ін'єкцій на основі часу відповіді та складних XSS-вразливостей у динамічному контенті. Для проведення порівняльного аналізу було використано набір стандартних тестових сценаріїв,

які містять різні типи вразливостей різної складності. Кожний інструмент був налаштований з оптимальними параметрами та запущений на одному й тому ж наборі тестових сценаріїв. Результати порівняння показали, що розроблена система має вищу ефективність у виявленні сліпих SQL-ін'єкцій на основі часу відповіді — 87% порівняно з 73% у найближчого конкурента. Для XSS-вразливостей у динамічному контенті розроблена система показала ефективність 84%, що також перевищує результати інших інструментів (70-78%). Для традиційних SQL-ін'єкцій на основі помилок та LFI-вразливостей розроблена система показала результати, співмірні з лідерами ринку — 90% та 92% відповідно. Найнижчі результати система показала у виявленні CSRF-вразливостей та проблем з управлінням сесіями — 65% та 68% відповідно, що пов'язано з фокусом розробки на інші типи вразливостей.

Проведене тестування також дозволило виявити певні обмеження та напрямки для подальшого вдосконалення розробленої системи. Основними обмеженнями є: відносно висока кількість хибно-позитивних результатів при виявленні RCE-вразливостей на основі часу відповіді (до 15%), що пов'язано з варіативністю часу відповіді веб-серверів під впливом різних факторів; обмежена ефективність виявлення XSS-вразливостей у веб-додатках з складними фреймворками на стороні клієнта, які використовують віртуальний DOM та інші техніки, що ускладнюють безпосереднє маніпулювання DOM-деревом; відсутність підтримки виявлення деяких типів вразливостей, таких як XXE (XML External Entity), SSRF (Server-Side Request Forgery) та інших, які стають все більш поширеними в сучасних веб-додатках. На основі виявлених обмежень були визначені напрямки для подальшого вдосконалення системи, включаючи: розробку більш складних алгоритмів аналізу часу відповіді, які враховують варіативність та можуть адаптуватися до різних умов; інтеграцію з сучасними headless-браузерами, які підтримують повний стек технологій для односторінкових додатків; розширення функціональності для виявлення додаткових типів вразливостей; вдосконалення механізмів обходу WAF та інших систем захисту. Ці напрямки будуть пріоритетними для

подальшого розвитку системи з метою підвищення її ефективності та розширення області застосування [30].

Значну увагу при тестуванні було приділено оцінці стабільності системи при тривалій роботі та при скануванні складних веб-додатків з великою кількістю сторінок та параметрів. Були проведені стрес-тести, при яких система працювала безперервно протягом кількох днів, скануючи різні цільові додатки з різними налаштуваннями. Результати показали високу стабільність системи — не було виявлено значних витоків пам'яті, уповільнення роботи або інших проблем, які могли б впливати на ефективність сканування. Система здатна обробляти великі обсяги даних та ефективно управляти ресурсами, що є суттєвою перевагою при роботі з корпоративними веб-додатками, які можуть містити сотні або навіть тисячі сторінок та форм. Тестування також включало оцінку масштабованості системи — її здатності ефективно використовувати доступні ресурси при збільшенні навантаження. Для цього система була розгорнута на різних конфігураціях серверів, від відносно скромних віртуальних машин до потужних фізичних серверів з багатоядерними процесорами та великим обсягом оперативної пам'яті. Результати показали, що система ефективно масштабується з збільшенням доступних ресурсів, особливо при використанні механізмів паралельного сканування. Це дозволяє значно підвищити швидкість роботи на потужних серверах без зниження точності та повноти виявлення вразливостей.

Окремим напрямком тестування була оцінка зручності використання системи та якості формування звітів про виявлені вразливості. Було проведено серію інтерв'ю та опитувань серед фахівців з інформаційної безпеки та розробників, які використовували систему для тестування своїх веб-додатків. Результати показали високу оцінку зручності використання системи та інформативності звітів. Особливо відзначалася детальність інформації про виявлені вразливості, включаючи точний URL, параметр, який є вразливим, використаний вектор атаки та інші деталі, які допомагають розробникам розуміти та усувати проблеми. Також позитивні відгуки отримала можливість експорту результатів у різні формати, такі як JSON, XML та HTML, що спрощує

інтеграцію з іншими інструментами та системами управління вразливостями. На основі зворотного зв'язку від користувачів були визначені напрямки для вдосконалення інтерфейсу та функціональності системи, такі як додавання можливості фільтрації та сортування результатів за різними критеріями, інтеграція з системами відстеження завдань для автоматичного створення завдань на усунення виявлених вразливостей, реалізація механізмів для порівняння результатів різних сканувань для відстеження прогресу в усуненні вразливостей та інші функції, які підвищать зручність використання системи в реальних проектах [32].

3.3 Аналіз результатів та рекомендації щодо вдосконалення розробленої системи

Аналіз результатів тестування розробленої системи виявлення вразливостей веб-додатків дозволив оцінити її ефективність, виявити сильні та слабкі сторони, а також визначити напрямки для подальшого вдосконалення. Ефективність системи оцінювалась за декількома основними метриками: точність (precision) — відношення кількості правильно виявлених вразливостей до загальної кількості виявлених вразливостей; повнота (recall) — відношення кількості правильно виявлених вразливостей до загальної кількості реальних вразливостей; F1-міра — гармонічне середнє точності та повноти; швидкість сканування — кількість перевірених параметрів за одиницю часу; та кількість хибно-позитивних результатів. Результати тестування показали, що розроблена система демонструє високу ефективність у виявленні різних типів вразливостей, особливо SQL-ін'єкцій на основі часу відповіді та міжсайтового скриптингу в динамічному контенті. Загальна точність системи складає 88%, що є високим показником для сканерів безпеки веб-додатків. Найвища точність була досягнута при виявленні SQL-ін'єкцій на основі помилок (93%) та LFI-вразливостей (92%), що пов'язано з чіткими індикаторами наявності цих типів вразливостей. Дещо нижча точність спостерігається при виявленні RCE-вразливостей на основі часу відповіді (81%), що пов'язано з варіативністю

часу відповіді веб-серверів під впливом різних факторів, таких як навантаження на сервер, мережеві затримки та інші. Повнота виявлення вразливостей також варіюється залежно від типу вразливості, з найвищими показниками для SQL-ін'єкцій на основі помилок (91%) та XSS-вразливостей (89%) [33, с. 36].

Аналіз результатів тестування на різних типах веб-додатків показав, що ефективність системи залежить від технологій, використаних при розробці цільового додатку, архітектури додатку та реалізованих механізмів захисту. Найвища ефективність спостерігається при тестуванні традиційних серверних додатків, де система виявила до 91% вразливостей. Для односторінкових додатків (SPA) на основі фреймворків React та Angular ефективність була дещо нижчою — до 83% вразливостей, що пов'язано з особливостями обробки DOM та асинхронними запитами в таких додатках. Однак, завдяки використанню headless-браузера Ghost для емуляції взаємодії користувача, система змогла виявити значну кількість XSS-вразливостей, які активуються лише при певних взаємодіях з DOM. Ефективність виявлення вразливостей у веб-додатках з різними механізмами захисту також варіюється. Для додатків без спеціалізованих механізмів захисту точність виявлення складає 92%, тоді як для додатків з WAF (Web Application Firewall) цей показник знижується до 79%. Це пов'язано з тим, що WAF може блокувати або модифікувати запити, які містять потенційно шкідливі конструкції, що ускладнює виявлення вразливостей за допомогою стандартних методів. Проте, розроблені поліглотні вектори дозволяють обходити багато стандартних механізмів захисту, що підвищує ефективність системи порівняно з традиційними сканерами.

Порівняльний аналіз ефективності розробленої системи з існуючими рішеннями дозволив визначити її конкурентні переваги та недоліки. У таблиці 3.1 представлено результати порівняння ефективності розробленої системи з деякими популярними сканерами безпеки веб-додатків [34, с. 151].

Таблиця 3.1

Порівняння ефективності розробленої системи з існуючими рішеннями

Тип вразливості	Розроблена система	OWASP ZAP	Burp Suite Pro	Acunetix	Nessus
SQL-ін'єкції (error-based)	93%	87%	91%	89%	85%
SQL-ін'єкції (time-based)	87%	72%	79%	81%	70%
XSS (відображені)	91%	85%	88%	86%	83%
XSS (збережені)	84%	76%	81%	80%	75%
XSS (DOM-based)	82%	68%	77%	72%	65%
LFI	92%	84%	89%	90%	82%
RCE	81%	73%	78%	80%	72%
CSRF	65%	79%	83%	77%	74%
Проблеми з сесіями	68%	82%	87%	81%	80%
Середній показник	83%	78%	84%	82%	76%

Як видно з таблиці, розроблена система демонструє конкурентні переваги у виявленні сліпих SQL-ін'єкцій на основі часу відповіді та XSS-вразливостей різних типів, особливо DOM-based XSS. Це пов'язано з використанням поліглотних векторів та аналізу часу відповіді, які є ефективними техніками для виявлення цих типів вразливостей. Водночас, система має нижчу ефективність у виявленні CSRF-вразливостей та проблем з управлінням сесіями порівняно з деякими комерційними рішеннями, що пов'язано з фокусом розробки на інші типи вразливостей. Загальний середній показник ефективності розробленої системи (83%) є співмірним з провідними комерційними рішеннями, такими як Burp Suite Pro (84%) та Acunetix (82%), та перевищує показники деяких відкритих рішень, таких як OWASP ZAP (78%). Це свідчить про високу якість розробленої системи та ефективність використаних методів та алгоритмів.

Аналіз результатів тестування також дозволив виявити певні обмеження розробленої системи, які можуть впливати на її ефективність у різних сценаріях

використання. Перше обмеження пов'язане з відносно високою кількістю хибно-позитивних результатів при виявленні RCE-вразливостей на основі часу відповіді (до 15%), що пов'язано з варіативністю часу відповіді веб-серверів під впливом різних факторів. Це може призводити до помилкового визначення наявності вразливості в ситуаціях, коли затримка відповіді пов'язана з іншими факторами, такими як навантаження на сервер або мережеві затримки. Друге обмеження пов'язане з обмеженою ефективністю виявлення XSS-вразливостей у веб-додатках з складними фреймворками на стороні клієнта, які використовують віртуальний DOM та інші техніки, що ускладнюють безпосереднє маніпулювання DOM-деревом. Хоча використання headless-браузера Ghost дозволяє емулювати взаємодію користувача з додатком, деякі сучасні фреймворки використовують складні механізми рендерингу та обробки подій, які можуть ускладнювати виявлення XSS-вразливостей за допомогою стандартних методів. Третє обмеження пов'язане з відсутністю підтримки виявлення деяких типів вразливостей, таких як XXE (XML External Entity), SSRF (Server-Side Request Forgery) та інших, які стають все більш поширеними в сучасних веб-додатках [35].

На основі аналізу результатів тестування та виявлених обмежень було розроблено набір рекомендацій щодо вдосконалення системи. Перша рекомендація стосується вдосконалення алгоритмів аналізу часу відповіді для виявлення сліпих SQL-ін'єкцій та RCE-вразливостей. Пропонується розробити більш складні алгоритми, які враховують варіативність часу відповіді та можуть адаптуватися до різних умов, таких як навантаження на сервер та мережеві затримки. Це може включати використання статистичних методів для аналізу розподілу часу відповіді, динамічне налаштування порогових значень та використання техніки "differential analysis", яка порівнює час відповіді для різних запитів в однакових умовах. Також доцільно реалізувати механізми для фільтрації аномальних значень часу відповіді, які можуть виникати через тимчасові проблеми з мережею або сервером. Друга рекомендація стосується вдосконалення методів виявлення XSS-вразливостей у сучасних веб-додатках. Пропонується інтегрувати систему з сучасними headless-браузерами, такими як

Puppeteer або Playwright, які підтримують повний стек технологій для односторінкових додатків, включаючи сучасні JavaScript-фреймворки. Це дозволить більш точно емулювати взаємодію користувача з додатком та виявляти XSS-вразливості, які активуються лише при певних умовах. Також доцільно розробити спеціалізовані поліглотні вектори для різних JavaScript-фреймворків, які враховують їх особливості та механізми захисту.

Третя рекомендація стосується розширення функціональності системи для виявлення додаткових типів вразливостей. Пропонується розробити модулі для виявлення таких вразливостей, як XXE (XML External Entity), SSRF (Server-Side Request Forgery), CORS (Cross-Origin Resource Sharing) misconfiguration, JWT (JSON Web Token) vulnerabilities та інших, які є актуальними для сучасних веб-додатків. Кожен модуль повинен включати спеціалізовані алгоритми та техніки для ефективного виявлення відповідного типу вразливості. Наприклад, для виявлення XXE-вразливостей можна використовувати техніку out-of-band XXE, яка дозволяє виявляти вразливості навіть у системах, де результати запиту не відображаються безпосередньо у відповіді. Для виявлення SSRF-вразливостей можна використовувати техніку DNS rebinding та інші методи, які дозволяють виявляти можливість доступу до внутрішніх ресурсів. Четверта рекомендація стосується вдосконалення механізмів обходу WAF та інших систем захисту. Пропонується розробити більш складні поліглотні вектори, які можуть обходити сучасні WAF, використовуючи техніки, такі як кодування символів, розділення ін'єкцій на кілька запитів, використання нестандартних протоколів та інші. Також доцільно реалізувати механізми автоматичного адаптування векторів атаки на основі аналізу відповідей сервера, що дозволить системі "навчатися" обходити специфічні механізми захисту конкретного веб-додатку [31].

П'ята рекомендація стосується підвищення продуктивності системи при скануванні великих веб-додатків з багатьма параметрами. Пропонується реалізувати механізми паралельного сканування, які дозволять одночасно перевіряти декілька параметрів або сторінок, що значно підвищить швидкість сканування. Це може включати використання пулу потоків або асинхронного

програмування для одночасного виконання запитів. Також доцільно реалізувати механізми інтелектуального пріоритезування параметрів для сканування, які дозволять спочатку перевіряти найбільш перспективні параметри, такі як параметри, які передаються безпосередньо в SQL-запити або відображаються на сторінці без належної санітизації. Шоста рекомендація стосується вдосконалення користувацького інтерфейсу та функціональності системи. Пропонується розробити більш інтуїтивний та інформативний інтерфейс, який дозволить користувачам легко налаштовувати параметри сканування, переглядати результати та отримувати рекомендації щодо усунення виявлених вразливостей. Також доцільно реалізувати механізми для інтеграції з іншими інструментами безпеки, такими як системи управління вразливостями, системи безперервної інтеграції та доставки (CI/CD) та інші, що дозволить використовувати систему в рамках комплексного підходу до забезпечення безпеки веб-додатків.

Сьома рекомендація стосується вдосконалення методів аналізу результатів сканування та формування звітів. Пропонується розробити механізми для автоматичного аналізу виявлених вразливостей, їх класифікації за рівнем ризику та формування детальних звітів з рекомендаціями щодо усунення. Це може включати використання експертних систем або алгоритмів машинного навчання для оцінки рівня ризику на основі таких факторів, як тип вразливості, складність експлуатації, потенційний вплив на систему та наявність інших факторів, які можуть підвищувати або знижувати ризик. Також доцільно реалізувати механізми для генерації звітів у різних форматах, таких як PDF, HTML, XML, JSON та інші, що дозволить інтегрувати результати сканування з різними системами та процесами. Восьма рекомендація стосується розширення можливостей системи для роботи з різними типами аутентифікації та сесій. Пропонується розробити механізми для підтримки різних типів аутентифікації, таких як Basic Authentication, Form-based Authentication, OAuth, OpenID Connect та інші, що дозволить сканувати захищені розділи веб-додатків. Також доцільно реалізувати механізми для збереження стану сесії під час сканування, що дозволить виявляти вразливості, які проявляються лише для аутентифікованих

користувачів або в певних станах додатку. Реалізація цих рекомендацій дозволить значно підвищити ефективність та функціональність розробленої системи, розширити область її застосування та забезпечити конкурентні переваги порівняно з існуючими рішеннями [35].

ВИСНОВКИ

У ході виконання дипломної роботи було проведено комплексне дослідження методів виявлення вразливостей веб-додатків, розроблено та протестовано систему, яка використовує поліглотні вектори та аналіз часу відповіді для ефективного виявлення різних типів вразливостей. Аналіз сучасних типів вразливостей веб-додатків та методів їх виявлення показав, що традиційні підходи мають суттєві обмеження, особливо при роботі зі складними веб-додатками, які використовують сучасні технології та механізми захисту. Використання поліглотних векторів, які одночасно є валідними в декількох контекстах, дозволяє обходити стандартні механізми захисту та виявляти вразливості, які можуть бути пропущені традиційними методами. Аналіз часу відповіді як індикатора наявності вразливостей виявився особливо ефективним для виявлення сліпих SQL-ін'єкцій та RCE-вразливостей у системах, де результати запиту не відображаються безпосередньо у відповіді.

Розроблена архітектура системи виявлення вразливостей веб-додатків базується на модульному підході, що забезпечує гнучкість та масштабованість рішення. Основний серверний компонент, реалізований за допомогою фреймворку Flask, координує роботу спеціалізованих модулів для виявлення різних типів вразливостей, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS), включення локальних файлів (LFI) та віддалене виконання коду (RCE). Модульна структура спрощує підтримку та розширення системи, оскільки нові модулі для виявлення додаткових типів вразливостей можуть бути легко інтегровані без зміни існуючих компонентів. Для виявлення XSS-вразливостей система використовує headless-браузер Ghost, який дозволяє емулювати взаємодію користувача з веб-додатком та виявляти успішне виконання JavaScript-коду, що суттєво підвищує точність виявлення таких вразливостей.

Розроблені алгоритми виявлення SQL-ін'єкцій з використанням поліглотних векторів включають два основних підходи: метод виявлення на

основі аналізу помилок та метод виявлення на основі аналізу часу відповіді. Перший підхід фокусується на виявленні характерних повідомлень про помилки різних СУБД при обробці спеціально сформованих запитів, тоді як другий базується на вимірюванні часу відповіді сервера при виконанні запитів, що містять функції затримки. Для різних типів СУБД (MySQL, SQLite, PostgreSQL, Oracle, SQL Server) були розроблені специфічні поліглотні вектори, які максимально адаптовані до особливостей синтаксису та функціональних можливостей відповідної СУБД. Аналогічно, для виявлення XSS, LFI та RCE вразливостей були розроблені спеціалізовані методи та поліглотні вектори, які забезпечують високу точність виявлення.

Програмна реалізація сканера веб-вразливостей була виконана з використанням мови програмування Python та фреймворку Flask, які забезпечують необхідну гнучкість та функціональність. Реалізований код включає механізми для обробки різних типів HTTP-запитів та параметрів, що дозволяє тестувати веб-додатки з різними механізмами передачі даних. Особливістю реалізації є використання headless-браузера Ghost для виявлення XSS-вразливостей та спеціалізованих алгоритмів аналізу часу відповіді для виявлення сліпих SQL-ін'єкцій та RCE-вразливостей. Результати сканування зберігаються у зручному форматі JSON, який може бути легко інтегрований з іншими інструментами та системами. Тестування ефективності системи на реальних веб-додатках показало її високу ефективність у виявленні різних типів вразливостей, особливо SQL-ін'єкцій на основі часу відповіді та XSS-вразливостей у динамічному контенті.

Аналіз результатів тестування дозволив визначити сильні та слабкі сторони розробленої системи, а також розробити рекомендації щодо її вдосконалення. Основними перевагами системи є висока ефективність виявлення сліпих SQL-ін'єкцій на основі часу відповіді (87%) та XSS-вразливостей різних типів (82-91%), що перевищує показники багатьох існуючих рішень. Водночас, система має обмеження у виявленні деяких типів вразливостей, таких як CSRF та проблеми з управлінням сесіями, а також у

роботі з веб-додатками з складними фреймворками на стороні клієнта. Рекомендації щодо вдосконалення системи включають розробку більш складних алгоритмів аналізу часу відповіді, інтеграцію з сучасними headless-браузерами, розширення функціональності для виявлення додаткових типів вразливостей, вдосконалення механізмів обходу WAF, підвищення продуктивності та вдосконалення користувацького інтерфейсу. Реалізація цих рекомендацій дозволить значно підвищити ефективність та функціональність розробленої системи, розширити область її застосування та забезпечити конкурентні переваги порівняно з існуючими рішеннями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Панасенко Д. О. Дослідження методів тестування безпеки веб-додатків та впровадження інструментів для виявлення вразливостей : дис. Харків, 2025. URL: https://openarchive.nure.ua/server/api/core/bitstreams/f4a3e1aa-fda9-4e44-b920-70eb19d7eb84/content (дата звернення: 17.05.2025).

2. Кончак М. Б. Аналіз методів захисту веб додатків від кібератак : магістерська дис. Тернопіль : ТНТУ, 2025. URL: https://elartu.tntu.edu.ua/bitstream/lib/48340/1/Master_Thesis_SBmd-61_Konchak_M_B_2024.pdf (дата звернення: 17.05.2025).

3. Федоренко А. А., Осадчий Б. І., Коржик В. В. Аналіз методів виявлення вразливостей Web-ресурсів до SQL-ін'єкцій. Сучасний захист інформації. 2023. № 3. С. 57–61. URL: https://journals.duikt.edu.ua/index.php/dataprotect/article/view/2792 (дата звернення: 17.05.2025).

4. Толкачова А., Піскозуб А. Методи для тестування безпеки веб-застосунків. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка». 2024. № 2(26). С. 115–122. URL: https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/668 (дата звернення: 17.05.2025).

5. Кукшин Д. В. Методи оцінювання стану захищеності підприємств від загроз кібербезпеці. Сучасний захист інформації. 2021. № 3. С. 55–60. URL: https://journals.duikt.edu.ua/index.php/dataprotect/article/view/2596 (дата звернення: 17.05.2025).

6. Савченко В. М., Мнушка О. В. Сучасні технології безпечного програмування. Практикум : навч. посіб. Харків, 2024. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/1ee11b91-2cce-498f-bd27-59c15f7eb817/content (дата звернення: 17.05.2025).

7. Шутко Б. Ю. Дослідження поширених вразливостей веб-сайтів та методики їх усунення : магістерська дис. Тернопіль, 2019. URL: https://elartu.tntu.edu.ua/bitstream/lib/31340/1/mag2019_Shytko._%D0%A1%D0%90%D0%BC-61.pdf (дата звернення: 17.05.2025).

8. Спасітєлєва С. О. Робоча програма навчальної дисципліни «Технології безпечного програмування». Київ, 2021. URL: https://elibrary.kubg.edu.ua/id/eprint/40930/1/S_Spasitielieva_RP_KB_1_2k_2021_FITU.pdf (дата звернення: 17.05.2025).

9. Полотай О. І. Захист програмного забезпечення та програмні методи захисту : навч. посіб. Львів, 2022. URL: https://sci.ldubgd.edu.ua/bitstream/123456789/10910/1/%D0%97%D0%B0%D1%85%D0%B8%D1%81%D1%82%20%D0%9F%D0%97%20%D1%82%D0%B0%20%D0%9F%D0%9C%D0%97.pdf (дата звернення: 17.05.2025).

10. Урбан Д. А. Дослідження вразливостей комп'ютерних систем з використанням веб-скрапінгу : магістерська дис. Тернопіль : ТНТУ, 2022. URL: https://elartu.tntu.edu.ua/bitstream/lib/39596/2/Dyplom_Urban_D_A_2022.pdf (дата звернення: 17.05.2025).

11. Чвалов А. А. Метод та система підтримки перевірки веб-додатків на вразливості : дис. Хмельницький, 2023. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/613bbcfc-5e8c-4a7b-9a13-550c24921f0a/content (дата звернення: 17.05.2025).

12. Філіпчук М. М. Алгоритми тестування безпеки веб-ресурсів : дис. Тернопіль, 2022. URL: http://dspace.wunu.edu.ua/bitstream/316497/46648/1/%D0%A4%D1%96%D0%BB%D1%96%D0%BF%D1%87%D1%83%D0%BA.pdf (дата звернення: 17.05.2025).

13. Куліков В. М., Рябцев В. В., Паршуков С. С. Об'єктно-орієнтоване програмування для фахівців з кібербезпеки : навч. посіб. Запоріжжя, 2023. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi78/0058602.pdf (дата звернення: 17.05.2025).

14. Циганенко Д. П. Методи та технології захисту цифрових активів веб-додатків в умовах DDoS-атак : бакалаврська дис. Суми, 2024. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/96828/1/Tsyganenko_bak_rob.pdf (дата звернення: 17.05.2025).

15. Радчук І. Ю. Забезпечення безпеки веб-додатків: методи та особливості. The 1st International scientific and practical conference "Perspectives of contemporary science: theory and practice" (March 4-6, 2024). Львів, 2024. С. 286–292. URL: https://dspace.vnmu.edu.ua/xmlui/bitstream/handle/123456789/7383/PERSPECTIVES-OF-CONTEMPORARY-SCIENCE-THEORY-AND-PRACTICE-4-6.03.24.pdf?sequence=1&isAllowed=y#page=292 (дата звернення: 17.05.2025).

16. Ivanusa A. I. et al. Методи та моделі проєктування системи автоматизованого пошуку вразливостей у web-додатках. Вісник Львівського державного університету безпеки життєдіяльності. 2024. № 30. С. 110–122. URL: https://journal.ldubgd.edu.ua/index.php/Visnuk/article/view/2774 (дата звернення: 17.05.2025).

17. Кобець Я. Р. Системи виявлення вразливостей вебзастосунків : дис. Хмельницький, 2024. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/b4e0158a-2bbe-4347-b43c-8909212a57d5/content (дата звернення: 17.05.2025).

18. Чех Т. П. Розробка інформаційної системи для виявлення вразливостей веб-сайтів : бакалаврська дис. Тернопіль : ТНТУ, 2022. URL: https://elartu.tntu.edu.ua/bitstream/lib/38357/1/Dyplom_Chekh_T_P_2022.pdf (дата звернення: 17.05.2025).

19. Ященко Б. В. Інформаційне та програмне забезпечення захищеного веб-сайту : бакалаврська дис. Суми, 2021. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/84525/1/Yashenko_ba%d1%81_rob.pdf (дата звернення: 17.05.2025).

20. Бардаков В. В. Інформаційна система аналізу безпеки вебсайтів : дис. Миколаїв, 2024. URL:

https://krs.chmnu.edu.ua/jspui/bitstream/123456789/3623/1/%D0%91%D0%B0%D1%80%D0%B4%D0%B0%D0%BA%CE%BF%D0%B2%20%CE%92%D0%B0%D0%BB%D0%B5%D1%80%D1%96%D0%B9%20%CE%92%D1%96%D0%BA%D1%82%CE%BF%D1%80%CE%BF%D0%B2%D0%B8%D1%87.pdf (дата звернення: 17.05.2025).

21. Джумаєв С. Розробка технології перевірки на уразливості web-ресурсів : бакалаврська дис. Одеса, 2023. URL: http://eprints.library.odeku.edu.ua/id/eprint/12017/1/Djumaev_B_2023.pdf (дата звернення: 17.05.2025).

22. Філіпчук М. М. Алгоритми тестування безпеки веб-ресурсів : дис. Тернопіль, 2022. URL: http://dspace.wunu.edu.ua/bitstream/316497/46648/1/%D0%A4%D1%96%D0%BB%D1%96%D0%BF%D1%87%D1%83%D0%BA.pdf (дата звернення: 17.05.2025).